Plant/controller optimization by convex methods

Niewoehner, Robert Jay

Monterey, California. Naval Postgraduate School

Approved for public release; distribution is unlimited.

PLANT/CONTROLLER OPTIMIZATION BY CONVEX METHODS

by

Robert Jay Niewoehner, Jr.
Lieutenant Commander, United States Navy
B.S., United States Naval Academy, 1981
M.S.E.E., The Johns Hopkins University, 1981

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN AERONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL

June, 1994

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b. OFFICE SYMBOL (*if applicable*) AA	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (*City, State, and ZIP Code*) Monterey, CA 93943	7b. ADDRESS (*City, State, and ZIP Code*) Monterey, CA 93943
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (*if applicable*)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (*City, State, and ZIP Code*)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION

11. TITLE (*Include Security Classification*)

Plant/Controller Optimization by Convex Methods

12. PERSONAL AUTHOR(S)

Robert Jay Niewoehner, Jr.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (*Year, Month, Day*)	15. PAGE COUNT
Doctoral Dissertation	FROM _____ TO _____.	June 1994	282

16. SUPPLEMENTARY NOTATION

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government.

17. COSATI CODES			18. SUBJECT TERMS (*Continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUB-GROUP	H-infinity control, multi-objective control, H2/H-infinity control, Linear M trix Inequalities, Control Power Requirements, Flight Control Optimizatic Plant Optimization

19. ABSTRACT (*Continue on reverse if necessary and identify by block number*)

This report presents results of a three phase effort to demonstrate the use of convex control design techniques eronautical applications. The first phase was the demonstration of a methodology by which classical aircraft contro lesign requirements could be translated into the weighting matrices for $\mathcal{H}_\infty$ controller synthesis. The second ph xtended that methodology to the design of mixed $\mathcal{H}_\epsilon$ /$\mathcal{H}_\infty$ controllers. The third phase considered the problen inimizing the size of aircraft control surfaces while meeting closed-loop dynamic performance requirements.

Control sizing is a critical element in the design of Reduced Static Stability (RSS) aircraft. Inadequate control po laces the vehicle in peril, while too much control power forfeits the benefits of RSS, resulting in poorer performar ncreased weight, increased cost, increased drag, and increased observability. Non-heuristic methods have been requi y which the physical configuration and the accompanying controller can be designed directly from the flying quali pecifications. The optimization of the surfaces should be done while searching over the set of all controllers wh ogether in closed-loop, satisfy the flying qualities requirements. This report presents a methodology which simulta ously optimizes both the physical configuration and the control system of a rigid body, using performance requireme which can be posed as Linear Matrix Inequalities.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Isaac I. Kaminer	22b. TELEPHONE (*Include Area Code*) (408) 656-2972 / 22c. OFFICE SYMBOL AA/KA

D FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS F
UNCLASSIFIED

ABSTRACT

This report presents results of a three phase effort to demonstrate the use of convex control design techniques in aeronautical applications. The first phase was the demonstration of a methodology by which classical aircraft controller design requirements could be translated into the weighting matrices for $\mathcal{H}_\infty$ controller synthesis. The second phase extended that methodology to the design of mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ controllers. The third phase considered the problem of minimizing the size of aircraft control surfaces while meeting closed-loop dynamic performance requirements.

Control sizing is a critical element in the design of Reduced Static Stability (RSS) aircraft. Inadequate control power places the vehicle in peril, while too much control power forfeits the benefits of RSS, resulting in poorer performance, increased weight, increased cost, increased drag, and increased observability. Non-heuristic methods have been required by which the physical configuration and the accompanying controller can be designed directly from the flying qualities specifications. The optimization of the surfaces should be done while searching over the set of all controllers which, together in closed-loop, satisfy the flying qualities requirements. This report presents a methodology which simultaneously optimizes both the physical configuration and the control system of a rigid body, using performance requirements which can be posed as Linear Matrix Inequalities.

TABLE OF CONTENTS

I. INTRODUCTION 1

A. SPECIFIC OBJECTIVES 3

B. REPORT ORGANIZATION 5

II. CONVEX OPTIMIZATION 7

A. DEFINITIONS 7

1. General 7

2. Linear Matrix Inequalities 9

3. Schur Complements 10

B. NUMERICAL ALGORITHMS FOR THE SOLUTION OF CONVEX OPTIMIZATION PROBLEMS 11

1. Ellipsoidal Algorithm 11

2. Interior Point Methods 14

III. THEORETICAL BACKGROUND 18

A. $\mathcal{H}_\infty$ CONTROL 19

1. The ∞ Norm 19

2. (Sub)Optimal $\mathcal{H}_\infty$ Output-Feedback Control by Riccati Methods 21

3. State-Feedback $\mathcal{H}_\infty$ Synthesis by Convex Methods 23

B. MIXED $\mathcal{H}_2$ / $\mathcal{H}_\infty$ CONTROL 26

1. Continuous Mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ Control 26

2. Discrete Mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ Control 30

C. POLE PLACEMENT BY CONVEX METHODS 34

D. ROBUSTNESS ANALYSIS BY STRUCTURED SINGULAR VALUES (μ) 35

IV. THE DESIGN OF AUTOLAND CONTROLLERS FOR CARRIER-BASED F-14 AIRCRAFT BY $\mathcal{H}_\infty$ AND $\mathcal{H}_2$ / $\mathcal{H}_\infty$ METHODS . . . 39

A. INTRODUCTION 39

B. PROBLEM STATEMENT 42

1. Airplane and Model Description 43

2. Problem Description 45

3. Design Requirements 47

4. Uncertainty Modeling 48

C. $\mathcal{H}_\infty$ CONTROLLER DESIGN 51

1. Synthesis Model 51

2. The Design Procedure 54

a. State-Feedback Design - Determining the W_1 and W_2 Weights 55

b. Improving the State-Feedback Design: Rate Feedback for Damping 56

c. Output Feedback Controller Design - Selecting the Measurement Noise Weights 58

d. Output Feedback Controller Design - Selecting the Process Noise Weights 60

e. Linear Simulation- Assessing the Controller Structure . . 61

3. Specification Compliance 62

4. μ Analysis 64

5. Nonlinear Simulation 65

6. $\mathcal{H}_\infty$ Design Conclusion 66

D. MIXED $\mathcal{H}_2$ / $\mathcal{H}_\infty$ CONTROLLER DESIGN 67

1. Problem Description and Design Requirements 67
2. Synthesis Model 67
3. The Design Procedure 70
4. Specification Compliance 71
5. μ-Analysis 72
6. Linear Simulation 73
7. Mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ Controller Design Conclusions 75
E. CONTRASTING THE $\mathcal{H}_\infty$ AND $\mathcal{H}_2$ / $\mathcal{H}_\infty$ DESIGN TOOLS . . 75
F. CONCLUSION 76
V. INTEGRATED AIRCRAFT/CONTROLLER DESIGN BY LINEAR MATRIX INEQUALITIES 77
A. INTRODUCTION AND PROBLEM MOTIVATION 77
B. PROBLEM DESCRIPTION 81
C. APPLYING DISTURBANCE REJECTION REQUIREMENTS . . 85
1. Special Case: Plant/Controller Optimization as a Generalized Eigenvalue Problem 86
2. General Case 87
3. Plant/Controller Optimization with Multiple Joint $\mathcal{H}_\infty$ Constraints 93
D. APPLYING JOINT DISTURBANCE REJECTION/ STABILIZATION REQUIREMENTS 95
E. INCLUDING MANEUVERABILITY REQUIREMENTS 99
1. Static Maneuverability Requirements 101
2. Dynamic Maneuverability Requirements 104
a. Dynamic Maneuverability Requirements: Closed-Loop Formulation 104

(1) An $\mathcal{H}_\infty$ Approach 105
(2) A Lyapunov Approach 114
b. Dynamic Maneuverability Requirements: Open-Loop Formulation 115
F. ACCOMODATING MODEL UNCERTAINTY 121
G. GENERAL COMMENTS 124
1. Interpreting the Results 124
2. The Example Problems 124
3. Limitations of the Methodology 125
4. Other Applications 125
H. FUTURE DIRECTIONS 126
1. Convexity Issues 126
2. Other Convex Performance Constraints 127
I. CONCLUSIONS 127
J. RECOMMENDATIONS 129
VI. CONCLUSIONS AND RECOMMENDATIONS 130
A. $\mathcal{H}_\infty$ DESIGN EXAMPLE CONCLUSIONS AND RECOMMENDATIONS 130
B. MIXED $\mathcal{H}_2$ / $\mathcal{H}_\infty$ CONTROL CONCLUSIONS AND RECOMMENDATIONS 131
C. PLANT/CONTROLLER OPTIMIZATION CONCLUSIONS AND RECOMMENDATIONS 132
APPENDIX A: ALGORITHMS FOR THE SOLUTION OF THE MIXED $\mathcal{H}_2$ / $\mathcal{H}_\infty$ CONTROLLER SYNTHESIS PROBLEMS . . 134
A. FINDING THE GRADIENTS OF MATRIX FUNCTIONALS . . 134
1. Derivative of an Eigenvalue of a Symmetric Matrix Functional 135

2. Derivative of a Matrix Inverse 135
B. CONTINUOUS TIME MIXED $\mathcal{H}_2$ / $\mathcal{H}_\infty$ CONTROLLER DESIGN 135
1. Numerical Solution of the Continuous Time State-Feedback Problem 136
2. Continuous Time $\mathcal{H}_2$ / $\mathcal{H}_\infty$ State-Feedback Synthesis Codes . 140
3. Continuous Mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ Output-feedback Synthesis Codes 147
C. DISCRETE TIME MIXED $\mathcal{H}_2$ / $\mathcal{H}_\infty$ CONTROLLER SYNTHESIS 149
1. Numerical Solution of the Discrete Time Full-Information Problem 149
2. Discrete Time $\mathcal{H}_2$ / $\mathcal{H}_\infty$ Full-Information Controller Synthesis Codes 153
3. Discrete $\mathcal{H}_\infty$ Output-Feedback Controller Synthesis Problem 163
4. Discrete $\mathcal{H}_2$ / $\mathcal{H}_\infty$ Output-Feedback Controller 165
D. Validation of the Ellipsoidal Codes 167
APPENDIX B: CONTROLLER DESIGN EXAMPLE SCRIPTS 168
A. NONLINEAR MODELS 168
1. Equation of Motion 168
2. Openloop Simulink Model 171
3. Synthesis Model Construction 171
4. Closed-Loop Analysis Models 176
B. $\mathcal{H}_\infty$ DESIGN SCRIPTS 176
C. MIXED $\mathcal{H}_2$ / $\mathcal{H}_\infty$ CONTROLLER DESIGN SCRIPTS 183
APPENDIX C: INTERIOR POINT CODES 186
A. GENERAL REMARKS 186
1. Posing a Basis 186
2. Modifications to the Original Code 187

3. The Principal Code 189
4. The Subroutines 190
B. PRACTICAL ISSUES 191
1. Determining a Feasible Initial Point 191
2. Practical Observations 192
C. MATLAB FUNCTION FILES 193
APPENDIX D: PLANT/CONTROLLER OPTIMIZATION CODES . . . 205
A. PLANT/CONTROLLER OPTIMIZATION FUNCTIONS 205
1. Plant and Controller Optimization for an $\mathcal{H}_\infty$ Performance Constraint 206
2. Plant and Controller Optimization for an $\mathcal{H}_\infty$ Performance Constraint at Multiple Flight Conditions 215
3. Joint $\mathcal{H}_\infty$ Pole Placement Plant/Controller Optimization . . . 220
4. Plant/Controller Optimization with a Joint $\mathcal{H}_\infty$ Static Maneuverability Specification 227
5. Plant/Controller Optimization with Dynamic Maneuverability Constraints 231
6. Plant/Controller Optimization with Robustness Constraints . 237
7. Joint $\mathcal{H}_2$ / Pole-Placement Plant/Controller Optimization . . 242
B. EXAMPLE SCRIPTS 249
1. Example 1- Optimal Vertical Tail at a Single Flight Condition 249
2. Example 2- Optimal Vertical Tail at Multiple Flight Conditions 251
3. Example Three- Optimal Vertical Tail for Joint $\mathcal{H}_\infty$ Pole-Placement Specification 253
4. Example Four- Optimal Vertical Tail for Joint $\mathcal{H}_\infty$ Static Moment Specification 254

5. Example Five/Six- Plant and Controller Optimization with Maneuvering Constraints 254
6. Example Seven- Plant and Controller Optimization with Maneuvering Constraints and Directed Thrust 258

REFERENCES 263
INITIAL DISTRIBUTION LIST 266

LIST OF TABLES

4.1 STATE-FEEDBACK DESIGN: WEIGHTS/RESULTING BANDWIDTHS 57

4.2 STATE-FEEDBACK DESIGN: WEIGHTS/RESULTING BANDWIDTHS 71

LIST OF FIGURES

2.1 Graphical Depiction of the Ellipsoidal Algorithm 13

3.1 Standard-Feedback Configuration. 18

3.2 The $\mathcal{H}_2$ / $\mathcal{H}_\infty$ Synthesis Framework. 27

3.3 Standard feedback configuration with uncertainty block. 36

4.1 Uncertainty Model 51

4.2 Synthesis Model 52

4.3 Broken Loop Controller Responses 58

4.4 Closed-Loop Command Responses 59

4.5 Sensor Responses 60

4.6 Output Feedback Broken-Loop Controller Responses 62

4.7 Output Feedback Closed-Loop Command Responses 62

4.8 Output Feedback Broken-Loop Nyquist Plot 63

4.9 Control Loop Gain Singular Values 64

4.10 Structured Singular Value Plot 65

4.11 Nonlinear Simulation Results 66

4.12 Mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ Synthesis Model 68

4.13 Broken-Loop Controller Responses 72

4.14 Closed-Loop Command Responses 73

4.15 Broken-Loop Nyquist Response 73

4.16 Open-Loop Singular Values 74

4.17 Closed-Loop Structured Singular Values 74

4.18 Linear Simulation Results 75

5.1 Optimization History for Tail Volume (Example One) 92

5.2 Optimization History Multiple Flight Conditions (Example Two) . . 95
5.3 Optimization History for Joint $\mathcal{H}_\infty$ / Pole-Placement (Example Three) 99
5.4 Open-loop Formulation for Maneuverability Constraints. 100
5.5 Closed-Loop Formulation for Maneuverability Constraints. 100
5.6 Optimization History for Vertical Tail Volume with Static Constraint (Example Four) 104
5.7 Optimization History for Longitudinal F-14 Problem with Multiple Initial Conditions (Example Five) 113
5.8 Broken-Loop Control Response for Longitudinal F-14 Problem (Example Five) 114
5.9 Optimization History for F-14 Problem with Open-Loop Constraint (Example Six) 118
5.10 Optimization History for F-14 Problem with Directed Thrust (Example Seven) 121
5.11 Linearized Uncertainty Model 123
B.1 Open-Loop Simulink Model 172
B.2 Non-Linear Plant and Actuators Block 173
B.3 Output Integrators Block 174
B.4 Closed-Loop Simulink Model 177
B.5 Controller Block 178
B.6 Output Integrator Block 178

ACKNOWLEDGMENT

This report represents the synthesis of a great many disciplines, and so I found myself looking all over the globe for support and answers to my questions. I am considerably indebted to Anton Stoorvogel, of Eindhoven University, and Laurent El Ghaoui, of ENSTA, France, for their enthusiastic counsel and tutelage. I could always count on both of them for prompt and thorough responses to my questions. Both demonstrated supreme grace in the face of frequently naive and foolish questions. Closer to home, I need to acknowledge and thank my advisor, Issac Kaminer for his direction and patient instruction. I shall always be impressed by Issac's breadth of understanding in the subtleties of control design and analysis. His background has given him a thrill for applications grounded in a grasp of theory that I cannot imagine ever attaining.

Finally, I must thank God for having so blessed me at home, in all that he's graciously given me there— three precious tender shoots to gently feed and water, that they might grow to be mighty oaks, and a Godly woman to be their mother and my wife.

A wife of noble character who can find? She is worth far more than rubies. Her husband has full confidence in her and lacks nothing of value... Her children arise and call her blessed; her husband also and he praises her:'Many women do noble things, but you surpass them all'.

I. INTRODUCTION

Aeronautical applications have provided much of both the motivation and resources for recent advances in the field of controls engineering. In the quest for ever improving performance, the field of aerodynamics matured to the point that only incremental gains were possible. Consequently, the thrust for improved performance turned instead to more innovative ways of controlling air vehicles— allowing for unstable open-loop dynamics, shrinking control surfaces, and eliminating mechanical command systems in favor of "fly-by-wire" systems. These innovations in controls permitted the industry to exploit innovation in aerodynamics. The multi-input multi-output (MIMO) nature of flight dynamics, which severely taxed the methods of classical control design, and the fiscal resources available as a consequence of the industry's vitality, fueled the development of the tools which for the moment are referred to as "Modern Control." Included in the list of the most recent tools available to the controls designer are controllers which are designed through the solution of convex optimization problems. The first general objective of this research was to demonstrate a methodology of how these theoretical advances can be implemented in aeronautical applications.

The second phase of the research then built upon the first, and capitalized on the very recent convergence of several technologies. As suggested above, the pursuit for performance has led both the commercial and military aircraft industries into the realm of Relaxed Static Stability (RSS) aircraft. The benefits include enhanced maneuverability, lower drag, lower weight, and lower cost. Reduced static stability is achieved, in large part, by shrinking or eliminating surfaces or physical features whose sole purpose is to provide either control power or static stability. Consider-

able industry and government attention has been focused on the question of how one quantifies the thresholds for satisfactory dynamic performance. A dramatic revision to the traditional flying qualities specifications, from MIL-8785C [Ref. 1] to MIL-1797 [Ref. 2] was principally in response to these types of issues. An ongoing NASA/Navy research effort has for several years been trying to quantify thresholds and metrics for satisfactory dynamic response [Ref. 3, 4]. Note, however, that the focus has been establishing metrics for satisfactory flying qualities (the dynamic behavior observed by the aircrew). One published research effort (not associated with a specific airframe) has concentrated on the task of translating the flying qualities requirements into the domain of vehicle and controls design [Ref. 5]. In this 1987 study, the absence of both an appropriate theoretical framework and the requisite optimization tools constrained the controls design approach to classical methods. The very recent convergence of three key technologies: (1) the theoretical formulation of many controls problems as convex or affine optimization problems, (2) the development of efficient numerical methods for the solution of convex or affine optimization problems, and (3) the computational capacity of modern engineering workstations to execute such routines, now permits the formulation of these vehicle and controller design problems to be posed as tractable constrained optimization problems. This report proposes a theoretical formulation and demonstrates a methodology by which not only a vehicle's control system, but the physical configuration of the vehicle itself, may be posed as a tractable constrained optimization problem.

In the pursuit of the above general objectives, the research effort was comprised of three projects. The first involved simply the use of available commercial designs tools. The second involved the creation of controller design tools based on recent theoretical formulations, and then their application to a simple design problem. Finally, the third area, the optimization of vehicle control power characteristics required the

development of a theoretical formulation, the creation of the appropriate design tools, and then their application to simple design examples to illustrate the viability of the methodology. Though these three efforts may only seem loosely related, each of the first two projects had elements which were critical to the subsequent project(s). The next section provides an overview of each project and its specific objectives.

A. SPECIFIC OBJECTIVES

The first problem was pure $\mathcal{H}_\infty$ design for the autoland control system of a F-14 aircraft. The F-14 is a carrier-based fighter manufactured by Grumman Corporation, and was selected for various examples within this study because of the unique configuration of its control surfaces. This effort had several specific objectives:

1. Investigate how $\mathcal{H}_\infty$ control could be utilized to incorporate the F-14's Direct Lift Control (DLC) in the autoland problem. DLC is a powerful aerodynamic control surface which can directly decrease or increase the lift generated by the wing by the symmetric deflection/ retraction of over-wing spoilers. The DLC is currently dormant in the F-14's autoland configuration.

2. Further develop and demonstrate a methodology whereby scalar weighting functions could be used to tune an $\mathcal{H}_\infty$ controller to meet classical performance requirements, including sensor bandwidths. This was an extension of the work of other authors [Ref. 6, 7, 8, 9, 10].

3. Introduce a methodology for the robustness analysis of nonlinear air vehicles.

The design and analysis tools used here were commercially available, and the principal contribution of this section was the demonstration of a methodology for their use. Though convex methods were not applied in this problem, the development of the methodology was necessary as a foundation for the second problem.

The second phase of the work was a mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ controller design example. Again the F-14 autoland controller design problem was chosen. The objectives of this phase included:

1. Development of the computational design tools which could solve the continuous and discrete time $\mathcal{H}_2$ / $\mathcal{H}_\infty$ controller design problem. These problems had previously been theoretically posed by Rotea, Khargonekar, and Kaminer as convex optimization problems [Ref. 11, 12].

2. Demonstrate a methodology for the use of mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ controllers. It was assumed at the outset that the methodology would be a derivative of the methodology demonstrated in the first design problem.

This phase of the work provided the modeling skills and convex optimization skills which were necessary for the pursuit of the final phase of the research.

With the first two projects complete, the foundations were in place to pursue the principal objective. The appropriate sizing of aerodynamic control surfaces is a current issue as the result of the trend towards Relaxed Static Stability (RSS) aircraft. The methodology in practice today is for the aerodynamic configuration designer to provide a controls designer with a configuration for which he is to design a controller that will hopefully satisfy the specified open-loop and closed-loop performance requirements. The controls designer only influences the configuration in the sense that if there is inadequate control power to achieve the desired flying qualities, the design is sent back to the configuration designer to provide more control power. Absent is a method by which the performance requirements can directly be translated into an optimal configuration along with an accompanying feasible controller. Given that many common performance specifications are convex, the question posed was: "Is it possible, to formulate a convex controller design problem in which not

just the controller, but the plant itself is optimized?" We will refer to this as the *plant/controller optimization problem.* The final phase of the research consequently had three objectives:

1. Determine a theoretical formulation for the plant/controller optimization problem.

2. Design the computational tools necessary to implement the proposed solution.

3. Create multiple design examples to illustrate and validate the proposed solution.

B. REPORT ORGANIZATION

The report is organized so as to separate the discussion of computation issues from the engineering issues. Consequently, the main body of the report exclusively discusses either theoretical issues or their applications. The computer codes and their relevant discussions are then found in the various appendices.

The main body begins with an overview of the tools and theory which were then applied in pursuit of the above objectives. Chapter II presents a short discussion of convex optimization, and outlines the two numerical algorithms which were used to solve the convex optimization problems which occur in the report. Chapter III then presents the theoretical controls background upon which the research drew. Most important is the outline of convex and affine expressions for various control design problems. Because of the similarity of the two problems, Chapter IV presents the details of both the $\mathcal{H}_\infty$ and mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ design problems. This includes a description of the problem to be solved, and the methodology for both the synthesis and analysis of the resulting controllers. In both cases, a simulation exercise was performed to verify that the controller demonstrated the desired characteristics. Chapter V then presents a methodology by which the plant/controller optimization

problem can be formulated as Linear Matrix Inequalities (LMI's). This formulation then permits the solution of the problem by convex methods. This chapter also includes a number of examples demonstrating how various types of specifications can be accommodated by this methodology. Finally, Chapter VI provides a summary of the conclusions and recommendations of the report.

The appendices form the balance of the report and include the final versions of the various computer codes used to generate the results found in the main body. MATLAB was used for all the programming, and so the codes are written either as function files, or m-file scripts. Appendix A presents the derivation and listing of the codes which were used to solve the continuous and discrete time $\mathcal{H}_2$ / $\mathcal{H}_\infty$ controller design problems. Appendix B provides the materials which supported the two F-14 design example problems. This includes the SIMULINK models used both to form the synthesis model and perform the nonlinear simulation. The scripts used to perform/analyze the design are also listed. Next, Appendix C presents a listing of the interior point codes which were used to solve those problems posed as LMI's. The original versions of these codes were written at the University of Michigan, and were provided by Professor Pramod Khargonekar. They were then substantially modified by this author to improve their numerical efficiency and reliability. Finally, Appendix D presents the function files and scripts which were used to support and illustrate the plant/controller optimization material of Chapter V.

II. CONVEX OPTIMIZATION

This chapter provides essential background material on the general classes of problems which were considered, and outlines the tools available for solving these problems. The first section reviews the foundational mathematical definitions. Familiarity with these terms and relationships is a prerequisite, as they occur repeatedly throughout the report, and in part define the scope of this report. Next, the second section provides a brief overview of the two numerical tools which were applied in solving the various example problems.

The following notational conventions will be observed in this report. Greek letters represent scalars or scalar valued functions (e.g. $\lambda \in \mathbf{R}$ or $\phi(x) : \mathbf{R}^n \rightarrow \mathbf{R}$). Lower case letters represent vectors (e.g. $x \in \mathbf{R}^n$), with a subscript i indicating the ith element. Lastly, uppercase letters represent either matrices or matrix valued functions (e.g. $Y \in \mathbf{R}^{n \times m}$ or $F(x)$). Pairs of subscripts on a matrix are the indices for a particular element of the matrix. A single subscript on a matrix indicates a particular matrix in a set of matrices. Additional notation will be introduced later, when flight dynamics conventions prevail.

A. DEFINITIONS

1. General

In general, the optimization problems considered during this research were of the form:

> *Given the vector space* $\mathbf{R}^n$, *and the scalar valued functions* $\phi(x) : \mathbf{R}^n \rightarrow \mathbf{R}$ *and* $\psi(x) : \mathbf{R}^n \rightarrow \mathbf{R}$, *find* $x_{opt} \in \mathbf{R}^n$, *such that* $\phi(x)$ *is minimized, subject to* $\psi(x) < 0$.

The function $\psi(x)$ is referred to as the *constraint function*, and if $\psi(s) < 0$, then $s \in \mathbf{R}^n$ is referred to as a *feasible* solution. The function $\phi(x)$ is referred to as either the *objective function* or *cost function*. In practice, a (sub)optimal search was performed to find a $x_{sub} \in \mathbf{R}^n$ such that $\phi(x_{sub}) - \phi(x_{opt}) < \nu$, where ν was an arbitrarily small stopping criteria.

The following mathematical definitions are important in describing various types of functionals. The definitions are extracted from [Ref. 13], but are standard across the literature. Consider the set $\mathbf{X} \in \mathbf{R}^n$.

Definition 2.1 *The set* $\mathbf{X}$ *is "affine," if for any* x, $\tilde{x} \in \mathbf{X}$ *and any* $\lambda \in \mathbf{R}$, $\lambda x + (1 - \lambda)\tilde{x} \in \mathbf{X}$.

Definition 2.2 *The set* $\mathbf{X}$ *is "convex," if for any* x, $\tilde{x} \in \mathbf{X}$ *and any* $\lambda \in [0, 1]$, $\lambda x + (1 - \lambda)\tilde{x} \in \mathbf{X}$.

Definition 2.3 *The functional* $\phi : \mathbf{X} \rightarrow \mathbf{R}$ *is "affine," if for any* x, $\tilde{x} \in \mathbf{X}$ *and any* $\lambda \in \mathbf{R}$, $\phi(\lambda x + (1 - \lambda)\tilde{x}) \leq \lambda\phi(x) + (1 - \lambda)\phi(\tilde{x})$.

Definition 2.4 *The functional* $\phi : \mathbf{X} \rightarrow \mathbf{R}$ *is "convex," if for any* x, $\tilde{x} \in \mathbf{X}$ *and any* $\lambda \in [0, 1]$, $\phi(\lambda x + (1 - \lambda)\tilde{x}) \leq \lambda\phi(x) + (1 - \lambda)\phi(\tilde{x})$.

Definition 2.5 *The functional* ϕ *on the convex set* $\mathbf{X}$ *is "quasi-convex ," if for any* x, $\tilde{x} \in \mathbf{X}$ *and any* $\lambda \in [0, 1]$, $\phi(\lambda x + (1 - \lambda)\tilde{x}) \leq max(\phi(x), \phi(\tilde{x}))$.

The following relationships can be deduced from the definitions:

1. an affine set is convex ,
2. an affine functional is convex ,
3. a convex functional is quasi-convex,
4. the reciprocal of an affine function is affine.

The most significant fact relating convex sets and (quasi)convex functionals is that if $\psi(x)$ is (quasi)convex, and $\alpha \in \mathbf{R}$, then the set $\mathbf{X}$ containing all x, such that $\psi(x) < \alpha$, is convex. Similarly, if ψ is affine and $\alpha \in \mathbf{R}$, then the set $\mathbf{X}$ containing all x, such that $\psi(x) < \alpha$, is affine. These are referred to as *functional inequality specifications.* The practical significance of quasi-convex functionals and the convex sets represented by a functional inequality specification is that one is guaranteed to find the global minimum of an objective function to within a numerical threshold. Furthermore, if the set is bounded, then the argument minimizing the objective function can also be isolated. The advantage of convex functionals over those that are quasi-convex is the facility with which lower bounds can be computed during the optimization process, resulting in straightforward termination criteria. The reference [Ref. 13] contains additional information on the properties of convex sets and functionals, as well as illustrations and alternative tests for convexity. The optimization problem described above is a (quasi)convex optimization problem if the set satisfying the constraining functional inequality specification is convex, and the objective function is (quasi)convex.

The optimization problem described above prescribed that the constraint functional $\psi(x)$ be scalar. Many of the constraint functionals encountered in this report will be matrix inequalities of the form: $H(x) = H^T(x) < 0$. This is mathematically equivalent to the scalar functional inequality: $\lambda_{max}(H(x)) < 0$. Consequently, we can use the functional matrix inequality to notationally represent the scalar constraint:$\lambda_{max}(H(x)) < 0$. Within all of the numerical algorithms applied here, it is the scalar constraint that is enforced.

2. Linear Matrix Inequalities

Consider the set of square, symmetric matrices $F_0, F_1, \ldots, F_n$, where $F_i = F_i^T \in \mathbf{R}^{m \times m}$, for all i.

Definition 2.6 *The functional inequality $F(x) > 0$, $x \in \mathbf{R}^n$, is a Linear Matrix Inequality (LMI), if it can be posed in the form $F(x) = F_0 + \sum_{i=1}^{n} x_i F_i > 0$.*

Note that the functional $\phi(x) = \lambda_{max}(-F(x))$ is affine. Consequently, the LMI , $F(x) > 0$, represents an affine functional inequality specification. Though the above mathematical definition of *affine* only pertains to scalar functions, it is extended it to include all matrix-valued functions of the form $F(x) = F_0 + \sum_{i=1}^{n} x_i F_i$. In this context, we refer to $F(x)$ as *affine* in x.

Several forms of optimization problems exist involving LMI's. The two problems of interest here are the Generalized Eigenvalue Problem (GEVP), and the Eigenvalue Problem(EVP). Let $A(x), B(x)$, and $C(x)$ be symmetric matrix-valued affine functions of x. The GEVP is defined as follows:

$$\begin{aligned} &\text{Minimize: } \lambda \\ &\text{Subject to: } \lambda B(x) - A(x) > 0, B(x) > 0, \text{ and } C(x) > 0. \end{aligned} \tag{2.1}$$

The EVP is the simplified case where $B(x) = I$. The important distinction between the two classes of problems is that the EVP is a convex optimization problem, while the GEVP is quasi-convex (see [Ref. 14]).

3. Schur Complements

The following lemma will be very helpful in reformulating various matrix inequalities.

Lemma 2.7 (Schur Complements) *Let Q, S, and R be matrices of compatible dimensions. Suppose $Q = Q^T$, and $S = S^T$. Then the following two statements are equivalent:*

1. $\begin{bmatrix} Q & R \\ R^T & S \end{bmatrix} > 0.$
2. $Q > 0, S > 0$, *and* $Q - RS^{-1}R^T > 0.$

Specifically, Schur complements are the means by which many of the Riccati inequalities common to modern control theory can be reformulated as LMI's.

B. NUMERICAL ALGORITHMS FOR THE SOLUTION OF CONVEX OPTIMIZATION PROBLEMS

Two numerical algorithms were used to solve the convex optimization problems of this research. At the point at which the research was undertaken, two principle numerical tools were available to pursue convex optimization problems: Kelly's cutting plane methods, and the Ellipsoid algorithm. The Ellipsoid algorithm was chosen both for it's ease of implementation, and its attributes regarding problem size. Kelly's cutting plane methods were rejected due to a concern about the growth of the data storage requirements for the size problems being considered. Shortly after the implementation of the Ellipsoidal codes for several problems, Interior Point methods began to mature and receive substantial attention in the controls community. These latter methods are applicable only to those convex optimization problems which can be posed as LMI's, but are reputed to converge much more quickly than the previous methods. In each case, Professor Stephen Boyd of Stanford University was the principal figure responsible for the popularization of these tools in the context of control theory applications.

This section provides a brief discussion of both of these methods. Only those details relevant to our specific implementation are addressed, as both these methods were regarded as means to an end. Details and convergence proofs, as well as further references regarding the history of these methods can be found in [Ref. 13, 14, 15].

1. Ellipsoidal Algorithm

The ellipsoidal algorithm is suitable for use in all quasi-convex optimization procedures, including LMI's, and is mathematically guaranteed to find a optimum

solution to within a specified threshold. First of all, consider a problem in which one is simply trying to optimize an objective function $\phi(x)$ without any constraints. Consider Figure 2.1. At each iteration k, the search is characterized by a vector $x^{(k)} \in \mathbf{R}^n$, and an ellipsoid $E^{(k)}$ centered about $x^{(k)}$, whose size and orientation are defined by the positive definite matrix $A^{(k)} \in \mathbf{R}^{n \times n}$ (the eigenvalues of $A^{(k)}$ are the square of the magnitudes of the respective semi-axes of $E^{(k)}$, while the eigenvectors of $A^{(k)}$ are their orientations). Assume that optimum point is located within $E^{(k)}$, and let the gradient vector g be defined: $g^{(k)} = \frac{\partial \phi(x)}{\partial x}\Big|_{x^{(k)}}$. If $\phi(x^{(k)}) = \alpha$, then $\phi(x)$ convex implies that $\phi(x) \geq \alpha$ for all x in that half of $E^{(k)}$ in the direction of $g^{(k)}$. The optimum point must then lie in the other half of $E^{(k)}$, and the half in the $g^{(k)}$ direction may be discarded from the search. Consequently, each iteration finds a new ellipsoid defined by $(x^{(k+1)}, A^{(k+1)})$, which completely contains the entire half-ellipsoid bounded by $((x^{(k)}, A^{(k)})$, and the hyper-plane orthogonal to $g^{(k)}$, in the $-g^{(k)}$ direction (the shaded area of Figure 2.1). The process is then repeated. Though the ellipsoid may elongate, the volume of the ellipsoid shrinks at a constant rate with each iteration, until the optimum point is isolated to suitable precision. Preferably, the ellipsoid should be initialized such that it includes the optimal point, though it is reportedly possible for the ellipsoid to migrate to capture the optimal point [Ref. 13].

This mechanism works equally well for constrained optimization problems. For the constrained problem, the new ellipsoid is used to either further isolate the feasible set defined by the constraint functional inequality, $\psi(x) < 0$, or reduce the cost function $\phi(x)$. The algorithm progresses as follows:

1. Evaluate $\psi(x^{(k)})$.

2. If $x^{(k)}$ is infeasible ($\psi(x^{(k)}) \geq 0$), then find the gradient $g_{\psi}^{(k)} = \frac{\partial \psi}{\partial x}\Big|_{x^{(k)}}$. By

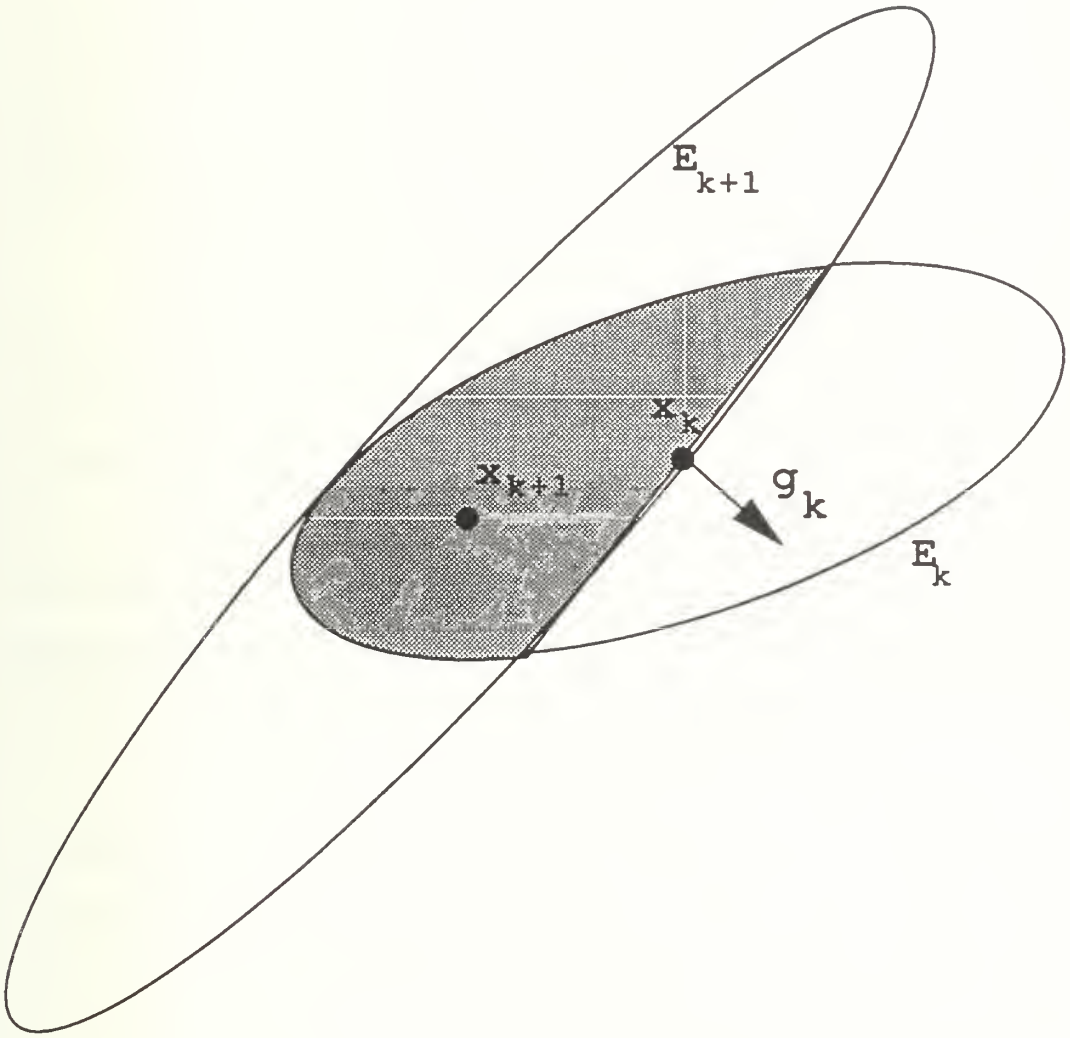

Figure 2.1: Graphical Depiction of the Ellipsoidal Algorithm

eliminating the infeasible half-space in the direction $g_{\psi}^{(k)}$, a new smaller ellipsoid $(x^{(k+1)}, A^{(k+1)})$ is determined. Provided that the search was initialized with a feasible point in the original ellipsoid, then all feasible points in $(x^{(k)}, A^{(k)})$ are retained in $(x^{(k+1)}, A^{(k+1)})$.

3. If $x^{(k)}$ is feasible $(\psi(x^{(k)}) < 0)$, then find the gradient $g_{\phi}^{(k)} = \frac{\partial \phi}{\partial x}\Big|_{x^{(k)}}$. Now the half space is eliminated for which the objective function has values greater than $(\phi(x^{(k)}))$, retaining in $(x^{(k+1)}, A^{(k+1)})$ all of the feasible points having objective values less than $(\phi(x^{(k)}))$.

4. Next iteration

Note that any number of constraint functions $\psi_i(x)$ could be considered sequentially in step 2 above. This structure is clearly apparent in the ellipsoidal codes in Appendix A (such as `h2infsyn`).

The various formulae for updating the ellipsoid can be found in the codes and in [Ref. 13]. The principal challenge in applying ellipsoidal methods was the derivation of the appropriate subgradients of functions which were not strictly differentiable.

In practice, a *deep-cut* modification to the above algorithm was used [Ref. 13]. The principal here was to use the value $\psi(x^{(k)}) > 0$ to shift the position of the hyper-plane in the $-g^{(k)}$ direction so as to reject more of the infeasible space with each iteration, improving the speed of convergence.

2. Interior Point Methods

Interior point methods for the efficient numerical solution of LMI's are generally attributed to Nesterov and Nemirovsky [Ref. 16]. Their application to problems of interest to the controls community was then popularized by Boyd and El Ghaoui [Ref. 14, 15].

Unlike the Ellipsoidal algorithm outlined above, the Interior Point algorithm is restricted to those optimization problems which can be posed as LMI's. Computationally, the interior point methods are superior to each of the other methods in part because the search is restricted to the feasible set (hence Interior Point), whereas both the Ellipsoidal and Cutting-Plane methods can exhaust tremendous amounts of computational energy on isolating the feasible set. Only a brief overview of the method is presented here, and the reader is referred to [Ref. 14, 15] for more thorough coverage.

Consider the EVP above. By including λ as the first element of the vector x (i.e. $x = [\ \lambda\ \ x^T\ ]^T$), and letting $c = [1, 0, \ldots, 0]$, the EVP can be reformulated:

$$\begin{aligned} &\text{Minimize: } \lambda = c^T x \\ &\text{Subject to: } F(x) := \begin{bmatrix} \lambda I - A(x) & \\ & C(x) \end{bmatrix} > 0. \end{aligned}$$

Let $\lambda^{(k)}$ represent an upper-bound on λ for iteration k. Let λ^{opt} represent the optimal value of the EVP, such that for all $\lambda^{(k)}) > \lambda^{opt}$, the LMI

$$\begin{bmatrix} \lambda^{(k)}) - c^T x & \\ & F(x) \end{bmatrix} > 0,$$

is feasible, i.e., there exists a vector x satisfying the LMI. If we assume that the LMI has a bounded feasible set, then the function

$$\phi^{(k)}(x) = \log\left(\det F(x)^{-1}\right) + \log \frac{1}{\lambda^{(k)} - c^T x} \tag{2.2}$$

has a global minimum within the bounded set since both terms are convex functions of x. The first term of $\phi(x)$ is a *boundary function* because its value goes to infinity as the boundary of the set $\{x :\ F(x) > 0\}$ is approached. It is this property that is used to keep the search within the feasible set. The choice of boundary function is not unique, and while it is unusual to find the determinant in a computational routine,

it is used here because both the gradient and Hessian of the boundary function are easily computed, and calculation of $\phi(x)$ itself is not required. The *analytic center* of $\phi(x, \lambda)$ is denoted as $x^*(\lambda^{(k)})$, and defined by:

$$x^*(\lambda^{(k)}) := \arg\min_x \left(\log \left(\det F(x, \lambda)^{-1} \right) + \log \frac{1}{\lambda^{(k)} - c^T x} \right). \tag{2.3}$$

The Interior Point method used here is based on the method of centers and is comprised of two nested loops. In the inner loop, given $\lambda^{(k)}$, the analytic center $x^*(\lambda^{(k)})$ can be found by Newton's method. In the outer loop, $\lambda^{(k)}$ is decreased with each iteration, and the search for $x^*(\lambda^{(k+1)})$ is initialized at $x^*(\lambda^{(k)})$. Algorithmically:

1. Initialize the problem at $k = 0$, with some feasible $x^{(0)}$ and $\lambda^{(0)}$, such that:

$$\begin{bmatrix} \lambda^{(0)}) - c^T x^{(0)} & \\ & F(x^{(0)}) \end{bmatrix} > 0,$$

2. Update $\lambda^{(k)}$:

$$\lambda^{(k+1)} = (1 - \theta) c^T x^{(k)} + \theta \lambda^{(k)}. \tag{2.4}$$

3. Find the analytic center $x^*(\lambda^{(k+1)})$ by Newton's method.

4. Update $x^{(k)}$:

$$x^{(k+1)} = x^*(\lambda^{(k+1)}), \tag{2.5}$$

5. Next k. Return to step 2 until termination criteria satisfied.

The variable $\theta \in (0, 1)$ is a computational parameter, with θ typically small. Note that the second term of $\phi(x)$ in 2.2 is singular if $\theta = 0$. In the outer loop, $x^{(k)}$ represents the set of analytic centers, which is described as the *path of centers*, hence the method is referred to as the *method of centers*.

Define the gradient (g) and Hessian (H) of ϕ to be:

$$\begin{aligned} g(x) &:= \frac{\partial \phi(x)}{\partial x} \quad \text{(a vector)} \\ H(x) &:= \frac{\partial^2 \phi(x)}{\partial x^2} \quad \text{(a matrix)}. \end{aligned}$$

The following algorithm outlines the Newton search for the analytic center, $x^*(\lambda^{(k)})$:

1. Initialize the Newton search with $x^{(k,1)} = x^{(k)}$.

2. Calculate each element of the gradient and Hessian of ϕ at $x^{(k,l)}$:

$$g_i(x^{(k,l)}) = \mathbf{tr}\left(F(x^{(k,l)})^{-1}F_i\right) + \frac{c_i}{\lambda^{(k)} - c^T x^{(k,l)}} \tag{2.6}$$

$$H_{ij}(x^{(k,l)}) = \mathbf{tr}\left(F(x^{(k,l)})^{-1}F_i F(x^{(k,l)})^{-1}F_j\right) + \frac{c_i c_j}{(\lambda^{(k)} - c^T x^{(k,l)})^2} \tag{2.7}$$

3. Determine the Newton decrement, δ, and the damping factor, α:

$$\delta(x^{(k,l)}) = \sqrt{g(x^{(k,l)})^T H(x^{(k,l)})^{-1} g(x^{(k,l)})} \tag{2.8}$$

$$\alpha(x^{(k,l)}) = \begin{cases} 1 & \text{if} \delta(x^{(k,l)}) \leq 0.25 \\ 1/(1+\delta(x^{(k,l)})) & \text{if} \delta(x^{(k,l)}) > 0.25 \end{cases} . \tag{2.9}$$

4. Update the search:

$$x^{(k,l+1)} = x(k,l) - \alpha(x^{(k,l)}) H(x^{(k,l)})^{-1} g(x^{(k,l)}). \tag{2.10}$$

5. Next l. Return to step 2, unless termination criteria satisfied.

6. $x^*(\lambda^{(k)}) = x^{(k,l)}$.

Details regarding the algorithm, including convergence proofs, termination criteria, and modifications, can be found in the above references. The interior point routines used for this research are discussed and documented in Appendix C. It is important to note that these routines are not problem specific but are suitable for solving any problem which has been posed as a GEVP (EVP). Once the problem is so posed, then the interior point algorithm requires only the three sets of basis matrices $\{A_0, A_1, \ldots, A_n\}$, $\{B_0, B_1, \ldots, B_n\}$, and $\{C_0, C_1, \ldots, C_n\}$.

III. THEORETICAL BACKGROUND

A wide range of interesting and powerful control problems can now be solved and applied as the result of recent advances in computational methods, computer capacity and theoretical control. The previous chapter outlined several of these computational tools. This chapter provides the theoretical background for the control design and analysis tools implemented in this report.

Consider the multiple input multiple output (MIMO) feedback system depicted in Figure 3.1. For the purposes of this chapter, we shall consider only finite dimensional linear time invariant systems. The general problem of control design is: Given

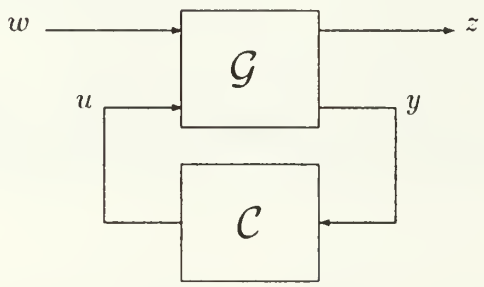

Figure 3.1: Standard-Feedback Configuration.

$\mathcal{G}$, find $\mathcal{C}$ such that the closed-loop system $\mathcal{T}(\mathcal{G}, \mathcal{C})$ is internally stable, and such that the output vector z has some specified desirable character in response to either the input vector w, or some specified set of initial conditions. It has long been recognized that many traditional control problems, such as solutions to Lyapunov's Equation, can be posed as a convex optimization problems. It is only with recent improvements in computational capacity (hard and soft), that these problems have become numerically tractable. Simultaneous with the increase in computational capacity came the

realization that many other desirable control problems could likewise be posed as convex optimization problems.

Initially, this chapter outlines several control problems which can be expressed as convex optimization problems. First, the $\mathcal{H}_\infty$ control problem will be defined and discussed. The pure $\mathcal{H}_\infty$ problem can be more rapidly solved by methods other than convex optimization, but its convex forms allow for several derivative problems of interest. Furthermore, it is the convex form that allows $\mathcal{H}_\infty$ to be used as the foundation for solutions to the plant/controller optimization problem. Next, the mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ control problem (continuous and discrete time) will be discussed and outlined. Third, a convex constraint for closed-loop pole locations will be presented.

After the presentation of the control design tools, the small gain theorems and the structured singular value will be discussed. By themselves, these are not design, but robustness analysis tools, and while not explicitly part of the convex optimization process, they can serve as a guide in formulating a synthesis model to design for robustness.

A. $\mathcal{H}_\infty$ CONTROL

1. The ∞ Norm

Consider again the feedback system depicted in Figure 3.1. Let $T_{zw}(\mathcal{G}, \mathcal{C})$ denote the closed-loop transfer function matrix from the input vector w of exogenous signals, to the output vector z of errors. Then the infinity norm of $T_{zw}(\mathcal{G}, \mathcal{C})$ is defined as the supremum over all frequencies of its largest singular value:

$$\| T_{zw}(\mathcal{G}, \mathcal{C}) \|_\infty := \sup_\omega \{\sigma(T_{zw}(j\omega))\},$$

where $\bar{\sigma}$ denotes the maximum singular value of T. This is an induced norm from w to z, which can alternatively be expressed,

$$\| T_{zw}(\mathcal{G}, \mathcal{C}) \|_\infty = \sup\{\| z \|_2 : \| w \|_2 \leq 1\}.$$

An interesting physical interpretation is that $\| T_{zw}(\mathcal{G}, \mathcal{C}) \|_\infty$ represents the peak power gain from w to z [Ref. 17]:

$$\| T_{zw}(\mathcal{G}, \mathcal{C}) \|_\infty = \sup \left\{ \frac{pow(z)}{pow(w)} \right\},$$

where:

$$pow(w) := \left(\lim_{T \to \infty} \frac{1}{2T} \int_{-T}^{T} w(t)^2 dt \right)^{1/2}.$$

The power interpretation clearly illustrates one of the many motivations behind the attention $\mathcal{H}_\infty$ work has received during the past decade. It is this property which will be exploited later, as specifications abound where a specified *rms* output level is permitted for a specified *rms* disturbance input. For example, the Dryden model of air turbulence specifies various levels of turbulence with *rms* amplitudes. This property of the $\mathcal{H}_\infty$ norm permits us to pose many typical disturbance rejection specifications as $\mathcal{H}_\infty$ control problems.

Consider the following state-space representation of the closed-loop system:

$$\mathcal{T}_{zw} := \left\{ \begin{array}{rcl} \dot{x} & = & Fx + Gw \\ z & = & Hx + Jw \end{array} \right., \tag{3.1}$$

where F is stable. It is well known [Ref. 18] that $\| T_{zw} \|_\infty < \gamma$, if and only if there exists a real symmetric matrix, $Y > 0$, such that:

$$FY + YF^T + (YH^T + GJ^T)(\gamma^2 I - JJ^T)^{-1}(HY + JG^T) + GG^T = 0. \tag{3.2}$$

This Riccati equation is referred to as the $\mathcal{H}_\infty$ analysis equation.

2. (Sub)Optimal $\mathcal{H}_\infty$ Output-Feedback Control by Riccati Methods

The $\mathcal{H}_\infty$ (sub)optimal control synthesis problem is to find, among all controllers that yield a stable closed-loop system, a controller $\mathcal{C}$ that minimizes $\| T_{zw}(\mathcal{G}, \mathcal{C}) \|_\infty$. Recent work [Ref. 18, 19, 20, 21, 22] has led to a simple and elegant approach to this problem.

Suppose that a continuous time state-space realization for the plant $\mathcal{G}$ depicted in Figure 3.1 can be written as

$$\mathcal{G} = \left\{ \begin{array}{rcl} \dot{x} & = & Ax + B_1 w + B_2 u \\ z & = & C_1 x + D_1 u \\ y & = & C_2 x + D_2 w \end{array} \right. . \tag{3.3}$$

Assume that (C_2, A, B_2) is stabilizable and detectable, that D_1 has linearly independent columns, and that D_2 has linearly independent rows. Recall that a *Hamiltonian matrix* is a matrix H of the form:

$$H = \left[\begin{array}{cc} P & R \\ Q & -P' \end{array} \right],$$

where P, Q and R are real $n \times n$ matrices with Q and R symmetric. If such a matrix H has no imaginary eigenvalues, then the spectral subspace $\lambda_-(H)$ spanned by the generalized eigenvectors belonging to eigenvalues lying in the open left half-plane is of dimension n. Let the columns of $\left[\begin{array}{c} X_1 \\ X_2 \end{array} \right]$ denote a basis for the subspace $\lambda_-(H)$. We will say that the Hamiltonian matrix H belongs to $dom(Ric)$ if H has no imaginary eigenvalues, and if X_1 is nonsingular. If H belongs to $dom(Ric)$, define $Ric(H) := X_2 X_1^{-1} =: X$. It is well known that X is symmetric, $P + RX$ is stable, and X satisfies the algebraic Riccati equation:

$$P'X + XP + XRX - Q = 0.$$

The key mathematical result on $\mathcal{H}_\infty$ synthesis by Riccati methods is stated below.

Theorem 3.1 *Consider the system 3.3. Suppose*

$$rank\left[\begin{array}{c|c} sI-A & B_2 \\ \hline -C_1 & D_1 \end{array}\right] = n + rank(D_1), \quad \forall s = j\omega.$$

and

$$rank\left[\begin{array}{c|c} sI-A & B_1 \\ \hline -C_2 & D_2 \end{array}\right] = n + rank(D_2), \quad \forall s = j\omega.$$

and let $\gamma > 0$ be a given positive number. Define

$$H(\gamma) = \begin{bmatrix} A - B_2(D_1'D_1)^{-1}D_1'C_1 & \gamma^{-2}B_1B_1' - B_2(D_1'D_1)^{-1}B_2' \\ -C_1'(I - D_1(D_1'D_1)^{-1}D_1')C_1 & -A' + C_1'D_1(D_1'D_1)^{-1}B_2' \end{bmatrix}$$

and,

$$J(\gamma) = \begin{bmatrix} A' - C_2'(D_2D_2')^{-1}D_2B_1' & \gamma^{-2}C_1'C_1 - C_2'(D_2D_2')^{-1}C_2 \\ -C_1(I - D_2'(D_2D_2')^{-1}D_2)B_1' & -A + B_1D_1(D_2D_2)^{-1}C_2 \end{bmatrix}$$

There exists a stabilizing controller $\mathcal{C}$ such that $\|T_{zw}\|_\infty < \gamma$ if and only if

1. *$H(\gamma) \in dom(Ric)$ and $X(\gamma) := Ric(H(\gamma))$ is positive semidefinite.*

2. *$J(\gamma) \in dom(Ric)$ and $Y(\gamma) := Ric(J(\gamma))$ is positive semidefinite.*

3. *$\rho(X(\gamma)Y(\gamma)) < \gamma^2$, where ρ denotes the spectral radius.*

Then such a controller, $\mathcal{C}$, is given by:

$$\mathcal{C} := \left\{ \begin{array}{rcl} \dot{\xi} & = & A_\infty\xi + Z_\infty Y_\infty C_2^T x \\ u & = & -B_2^T X_\infty \xi \end{array} \right. , \tag{3.4}$$

where

$$A_\infty := A + \gamma^{-2}B_1B_1^TX_\infty - B_2B_2^TX_\infty - Z_\infty Y_\infty C_2^TC_2 \tag{3.5}$$

$$Z_\infty := (I - \gamma^{-2}Y_\infty X_\infty)^{-1}. \tag{3.6}$$

Existence and computation of $X(\gamma)$ and $Y(\gamma)$ are standard matrix algebra problems that can be solved using a standard technique for solving Riccati equations

based on the real Schur decomposition [Ref. 23]. The pair of Riccati equations associated with $H(\gamma)$ and $J(\gamma)$ are referred to as the $\mathcal{H}_\infty$ synthesis equations. Specifically, the Riccati equation associated with $H(\gamma)$ is the state-feedback synthesis equation, while $J(\gamma)$ is associated with is the $\mathcal{H}_\infty$ filtering equation.

Commercial software is available from several sources which implements this theorem to determine a suitable controller from the input state-space model [Ref. 24]. In practice, implementations of this theorem usually start with an arbitrary upper bound γ_u on the achievable performance. The theorem is then used to perform a binary search in the interval $[0,\ \gamma_u]$ for the optimal value of γ. If γ_u proves infeasible, then it can be set arbitrarily higher. Once the binary search has determined a sufficiently small interval in which the optimal value of γ must lie, the search is stopped and a (sub)optimal controller $\mathcal{C}$ is computed using the right endpoint of this interval for γ in the formulae above. Controllers determined by this means are usually referred to as the *central controller*. These methods are not perfectly clean numerically, as in practice, X and Y must be allowed to have very small negative eigenvalues.

3. State-Feedback $\mathcal{H}_\infty$ Synthesis by Convex Methods

In this section, extracted largely from [Ref. 25], we show that the fractional representation of memoryless (i.e., static) state-feedback controllers, i.e. $K = WY^{-1}$, where $Y > 0$, can be used to reduce the state-feedback $\mathcal{H}_\infty$ control optimization problem to a convex feasibility problem over the space of finite-dimensional real matrices. This fractional representation was first introduced by [Ref. 26], and will be used extensively throughout this report.

In order to introduce this parameterization, we first answer the following question. Given a plant, $\mathcal{G}$, with all the states available for feedback, characterize the set $\mathcal{A}_m(\mathcal{G}\ )$ of all stabilizing memoryless state-feedback controllers. Suppose the

plant $\mathcal{G}$ is represented by the following equations

$$\mathcal{G} = \begin{cases} \dot{x} &= Ax + B_1 w + B_2 u \\ z &= C_1 x + D_1 u \\ y &= x, \end{cases} \tag{3.7}$$

where $x \in R^n$, $u \in R^q$ and $z \in R^p$. Let Σ denote the set of all real $n \times n$ symmetric matrices, and define

$$\Omega := \left\{ (W, Y) \in R^{q \times n} \times \Sigma : Y > 0 \right\} \tag{3.8}$$

Note that Ω is a strictly convex open subset of $R^{q \times n} \times \Sigma$

We now make the following assumption:

*A*1. The pair (A, B_2) is stabilizable.

Assumption *A*1 is necessary to guarantee that $\mathcal{A}_m(\mathcal{G})$ is not empty.

Theorem 3.2 *Let $\mathcal{G}$ be given by equation (3.7). Define*

$$L(W, Y) := AY + YA' + B_2 W + W' B_2'. \tag{3.9}$$

Consider the set

$$\Phi_m := \{(W, Y) \in \Omega : L(W, Y) < 0\}.$$

Then $\mathcal{A}_m(\mathcal{G})$ is nonempty if and only if Φ_m is nonempty. In this case Φ_m is convex and the mapping

$$\varphi : \Phi_m \to \mathcal{A}_m(\mathcal{G}) : (W, Y) \mapsto WY^{-1}$$

is onto.

Proof. Suppose $K \in \mathcal{A}_m(\mathcal{G})$. Then it follows from Liapunov stability theory that there exists $Y > 0$ such that

$$(A + B_2 K)Y + Y(A + B_2 K)' < 0. \tag{3.10}$$

Set $W = KY$ in (3.10) to get (3.9).

Conversely, suppose $(W, Y) \in \Phi_m$. Set $K = WY^{-1}$ in (3.9) to get (3.10).

Finally, convexity of Φ_m follows from the convexity of the mapping $Y \rightarrow L(Y)$, which is linear and, hence, affine and convex. ■

Next, we proceed to similarly parametrize the set of all memoryless stabilizing state-feedback controllers, which also make the infinity norm of the closed-loop transfer function

$\|T_{zw}(G, K)\|_\infty$ less than a given number $\gamma > 0$. We denote this set $\mathcal{A}_{\infty,m}(\mathcal{G})$. The importance of this theorem is in the fact that it parametrizes $\mathcal{A}_{\infty,m}(\mathcal{G})$ in terms of a set of solutions to a convex QMI (quadratic matrix inequality).

Theorem 3.3 *Let $\mathcal{G}$ be given by equation (3.7) and let $\gamma > 0$. Define*

$$R(W, Y) := AY + YA' + B_2W + W'B_2' + (C_1Y + D_1W)'(C_1Y + D_1W) + B_1B_1'/\gamma^2. \quad (3.11)$$

Consider the set

$$\Phi_{m,\infty} := \{(W, Y) \in \Omega : R(W, Y) < 0\}.$$

Then $\mathcal{A}_{\infty,m}(\mathcal{G})$ is nonempty, if and only if $\Phi_{m,\infty}$ is nonempty. In this case $\Phi_{m,\infty}$ is convex and the mapping

$$\varphi : \Phi_{m,\infty} \rightarrow \mathcal{A}_{\infty,m}(\mathcal{G}) : (W, Y) \mapsto WY^{-1}$$

is onto.

Proof. Without loss of generality assume $\gamma = 1$. Suppose $K \in \mathcal{A}_{\infty,m}(\mathcal{G})$ is given. Then, it follows that there exists $Y > 0$ such that the $\mathcal{H}_\infty$ analysis inequality 3.2 is satisfied:

$$(A + B_2K)Y + Y(A + B_2K)' + Y(C_1 + D_1K)'(C_1 + D_1K)Y + B_1B_1' < 0. \quad (3.12)$$

Set $W = KY$ in (3.12) to get (3.11)

Conversely, suppose $(W, Y) \in \Phi_{m,\infty}$ are given. Set $K = WY^{-1}$ to get (3.12). Then it follows that $K \in \mathcal{A}_{\infty,m}(\mathcal{G})$.

The convexity of $\Phi_{m,\infty}$ is proved in [Ref. 11]. ■

By Schur complements, inequality 3.11 is equivalent to the following LMI [Ref. 15, 20]:

$$R_1(W,Y) := \begin{bmatrix} AY + YA' + B_2W + W'B_2' + B_1B_1' & (CY + DW)' \\ (CY + DW) & -\gamma^2 I \end{bmatrix} < 0. \quad (3.13)$$

This LMI expression is superior to the QMI 3.11 in two respects. First of all, it is affine in the controller parameters W and Y, Secondly, the LMI is also jointly affine in γ^2. Consequently, while Riccati methods can only find a controller for a specified γ, and methods such as bisection are required to find the optimal γ, convex methods using the LMI expression can find a (sub)optimal γ and associated controller directly. Similar expressions exist for the output feedback synthesis problem [Ref. 27]. This LMI will figure critically in our formulation of the control power optimization problem.

B. MIXED $\mathcal{H}_2$ / $\mathcal{H}_\infty$ CONTROL

Multi-objective control, and most specifically, mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ control has received considerable attention in the controls literature in the past several years (see [Ref. 28, 12]) and references therein). This section presents the theoretical results which provide for the convex solution of the mixed problem in both continuous and discrete time cases. These results are largely extracted from [Ref. 28, 12].

1. Continuous Mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ Control

Consider the finite-dimensional linear time invariant (FDLTI) system depicted by Figure 3.2. The objective of the underlying problem is, given $\mathcal{G}$, find $\mathcal{C}$, such that the generalized $\mathcal{H}_2$ cost of the closed-loop system, $\|T_{z_0w}(\mathcal{G},\mathcal{C})\|_{2_f}$ is minimized, subject to the constraint that $\|T_{z_1w}(\mathcal{G},\mathcal{C})\|_\infty < \gamma$. The generalized $\mathcal{H}_2$ cost is defined as:

$$\|T_{zw}\|_{2_f} := \sqrt{f\left(\frac{1}{2\pi}\int_{-\infty}^{\infty} T_{zw}(j\omega)T_{zw}^*(j\omega)d\omega\right)}, \quad (3.14)$$

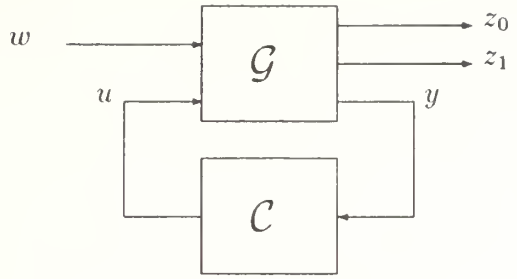

Figure 3.2: The $\mathcal{H}_2$ / $\mathcal{H}_\infty$ Synthesis Framework.

where $f(\cdot)$ can be either 1) the trace, 2) the maximum eigenvalue , or 3) the maximum diagonal element. The reader is referred to [Ref. 29] for a discussion of the attributes of these norms.

Suppose $\mathcal{A}_{\infty,m}(\mathcal{G})$ is nonempty, i.e., a controller, $\mathcal{C}$, exists such that $\|T_{z_1 w}(\mathcal{G},\mathcal{C})\|_\infty < \gamma$. The set defined by $\mathcal{A}_{\infty,m}(\mathcal{G})$ is then referred to as the set of *feasible* controllers. In [Ref. 28, 30], the authors introduce an associated cost, the mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ norm ($\|\cdot\|_{2/\infty}$), which they prove is an upper bound to the generalized $\mathcal{H}_2$ cost. The $\mathcal{H}_2$ / $\mathcal{H}_\infty$ problem can then be considered as a search over the set of feasible controllers for that controller which minimizes $\|T_{z_0 w}(\mathcal{G},\mathcal{C})\|_{2/\infty}$. Furthermore, the determination of a controller, $\mathcal{C}$, which minimizes $\|T_{z_0 w}(\mathcal{G},\mathcal{C})\|_{2/\infty}$, subject to $\|T_{z_1 w}(\mathcal{G},\mathcal{C})\|_\infty < \gamma$, is posed as the following convex optimization problem.

Consider the following state-space representation of the closed-loop system $T_{zw}(\mathcal{G},\mathcal{C})$, where $\mathcal{C}$ is feasible:

$$\mathcal{G} := \begin{cases} \dot{x} &= Fx + Gw \\ z_0 &= H_0 x + J_0 w \\ z_1 &= H_1 x + J_1 w \end{cases} \tag{3.15}$$

We know from equation 3.2 that there exists a unique symmetric matrix Y, such that:

$$FY + YF^T + (YH_1^T + GJ_1^T)(\gamma^2 I - J_1 J_1^T)^{-1}(H_1 Y + J_1 G^T) + GG^T = 0.$$

Moreover, if L_c is the controllability Grammian of the pair (F,G), then $0 \le L_c \le Y$.

The *generalized mixed* $\mathcal{H}_2$ / $\mathcal{H}_\infty$ *cost* of the closed loop system is then defined as:

$$J_f(T_{z_0 w}) := \begin{cases} \infty & \text{if } J_0 \neq 0 \\ f(H_0 Y H_0^T) & \text{otherwise} \end{cases} . \tag{3.16}$$

Consider now the following state-space representation of $\mathcal{G}$:

$$\mathcal{G} := \begin{cases} \dot{x} &= Ax + B_1 w + B_2 u \\ z_0 &= C_{01} x + D_{01} u \\ z_1 &= C_{11} x + D_{11} u \\ y &= C_2 x + D_{21} w \end{cases} . \tag{3.17}$$

Let $n := dim(x)$, and $m := dim(u)$. Furthermore, assume that: (a) the triple (C_2, A, B_2) is stabilizable and detectable, (b) the pair (A, B_1) has no uncontrollable modes on the imaginary axis, and (c) $D_{21}[B_1^T \;\; D_{21}^T] = [0 \;\; I]$.

Suppose there exists a symmetric matrix $Q > 0$, satisfying the $\mathcal{H}_\infty$ filtering equation:

$$AQ + QA^T + Q(C_1^T C_1 - C_2^T C_2)Q + B_1 B_1^T = 0. \tag{3.18}$$

Define the auxiliary quantities:

$$A_Q = A + QC_1^T C_1 \quad B_{1Q} := QC_2^T \quad B_{2Q} := B_2 + QC_1^T D_{11}$$

Let $(W, Y) \in \{W, Y : W \in \mathbf{R}^{m \times n}, Y \in \mathbf{R}^{n \times n}, Y = Y^T > 0\}$. Finally, define:

$$\begin{aligned} R(W,Y) &:= A_Q Y + Y A_Q^T + B_{2Q} W + W^T B_{2Q}^T + B_{1Q} B_1 1 Q^T + \\ &\qquad (C_1 Y + D_{12} W)^T (C_1 Y + D_{12} W) \end{aligned} \tag{3.19}$$

$$M(W,Y) := C_0 Q C_0^T + (C_0 Y + D_{02} W) Y^{-1} (C_0 Y + D_{02} W)^T. \tag{3.20}$$

Note that equation 3.19 is identical to 3.11.

Let Φ define the set of feasible solutions:

$$\Phi := \left\{ (W, Y) : Y = Y^T > 0, R(W, Y) < 0 \right\}, \tag{3.21}$$

and consider the optimization problem

$$\sigma_f := \inf \{ f(M(W, Y)) : (W, Y) \in \Phi \} . \tag{3.22}$$

Theorem 3.4 *The set of feasible controllers, $\mathcal{A}_{\infty,m}(\mathcal{G})$, is not empty, if and only if: (1) A stabilizing solution $Q \geq 0$ exists to 3.18, and (2) the set Φ is not empty. If a feasible controller exists, then*

$$J_f(T_{z_0w}(\mathcal{G}, \mathcal{C})) := f(M(W,Y)).$$

Moreover, given any $\alpha > \sigma_f$, there exists a solution , $(W,Y) \in \Phi$ such that $f(M(W,Y)) < \alpha$, and the dynamic observer based controller

$$\mathcal{C} := \begin{cases} \dot{\xi} &= A_Q\xi + B_{2Q}u + B_{1Q}(y - C_2\xi) \\ u &= WY^{-1}\xi \end{cases} \tag{3.23}$$

is feasible and $J_f(T_{z_0w}(\mathcal{G}, \mathcal{C})) < \alpha$.

Consequently, if a feasible controller exists, then the problem of finding a (sub) optimal controller for the $\mathcal{H}_2$ / $\mathcal{H}_\infty$ problem can be reduced to the solution of a Riccati equation (3.18), and the numerical solution to the convex optimization problem:

Minimize: $f(M(W,Y))$, over $(W,Y) \in \mathbf{R}^{m\times n} \times \mathbf{R}^{n\times n}$.

Subject to: $R(W,Y) < 0$, and $Y = Y^T > 0$.

The solution to the state-feedback problem ($C_2 = I, D_{21} = 0$) can be simply posed by eliminating consideration of the filtering equation (3.18), and replacing $A_Q = A$, $B_{1Q} = B_1$, and $B_{2Q} = B_2$ in the expressions for $R(W,Y)$ and $M(W,Y)$ above. The controller is then $\mathcal{C} = K_{sfb} = WY^{-1}$.

The ellipsoidal method was chosen to code a MATLAB function file which solved the state-feedback design problem. A second routine was also written which solved the measurement-feedback problem by solving the filtering equation (3.18) by Riccati methods, replacing the state-feedback variables with the appropriate auxiliary plant variables, and then calling the state-feedback design function. Both of these codes and their supporting subroutines are developed and then outlined in Appendix A.

It is a straightforward matter to use Schur complements to pose each of the above matrix valued inequalities, 3.19 and 3.20 as LMI's. Consequently, the numerical problem could also have been solved by the interior point method.

2. Discrete Mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ Control

This section presents the theoretical results of [Ref. 12], which pose the discrete $\mathcal{H}_2$ / $\mathcal{H}_\infty$ controller design problem as a convex optimization problem. The discrete time $\mathcal{H}_2$ / $\mathcal{H}_\infty$ problem differs only slightly from the continuous time problem above. The most significant difference is that it is structured about full-information feedback for an auxiliary plant rather than a state-feedback solution.

First of all, the definitions of the respective norms are predictably similar to their continuous time counterparts. The discrete time version of the generalized $\mathcal{H}_2$ norm is:

$$\|T_{zw}\|_{2/f} := \sqrt{f(\frac{1}{2\pi}\int_0^{2\pi}(T_{zw}T_{zw}^*)(e^{j\theta})\,d\theta)}, \tag{3.24}$$

where the function $f(\cdot)$ is again either the trace function, the maximum eigenvalue (λ_{max}), or the maximum diagonal entry (d_{max}). The $\mathcal{H}_\infty$ norm in discrete time can be defined as:

$$\|T_{zw}\|_\infty := \max_{\theta\in[0\ 2\pi]} \sigma_{max}(T_{zw}(e^{j\theta})), \tag{3.25}$$

where σ_{max} denotes the maximum singular value.

Suppose the plant, $\mathcal{G}$, in Figure 3.2 is now represented by the following discrete time state-space model:

$$\mathcal{G} := \left\{\begin{array}{rcl} \sigma x &=& Ax + B_1 w + B_2 u \\ z_0 &=& C_0 x + D_{01} w + D_{02} u \\ z_1 &=& C_1 x + D_{11} w + D_{12} u \\ y &=& C_2 x + D_{21} w, \end{array}\right. \tag{3.26}$$

where σ denotes the shift operator $(\sigma x)(k) := x(k+1)$. Assume that (1) the triple (C_2, A, B_2) is stabilizable and detectable, and (2) given any complex number z satis-

fying $|z| = 1$, the matrix

$$\begin{bmatrix} A - zI & B_1 \\ C_2 & D_{21} \end{bmatrix},$$

has full row rank. The solution to the output feedback problem makes use of a suitably constructed optimization problem defined next.

Suppose a feasible controller exists such that $\|T_{z1w}\|_\infty < 1$. Then, it follows from [Ref. 31] that there exists a (unique) real symmetric matrix $Q \geq 0$ such that the matrices V and R defined by

$$V := C_2 Q C_2^T + D_{21} D_{21}^T, \tag{3.27}$$

$$R := I - D_{11} D_{11}^T - C_1 Q C_1^T + (C_1 Q C_2^T + D_{11} D_{21}^T) V^{-1} (C_1 Q C_2^T + D_{11} D_{21}^T)^T, \tag{3.28}$$

are positive definite. Moreover, Q satisfies the following discrete algebraic Riccati equation:

$$Q = A Q A^T + B_1 B_1^T - \begin{bmatrix} C_2 Q A^T + D_{21} B_1^T \\ C_1 Q A' + D_{11} B_1^T \end{bmatrix}^T P(Q)^{-1} \begin{bmatrix} C_2 Q A^T + D_{21} B_1^T \\ C_1 Q A^T + D_{11} B_1^T \end{bmatrix}, \tag{3.29}$$

where

$$P(Q) := \begin{bmatrix} D_{21} D_{21}^T & D_{21} D_{11}^T \\ D_{11} D_{21}^T & D_{11} D_{11}^T - I \end{bmatrix} + \begin{bmatrix} C_2 \\ C_1 \end{bmatrix} Q [C_2^T \; C_1^T],$$

and the matrix

$$A - \begin{bmatrix} C_2 Q A^T + D_{21} B_1^T \\ C_1 Q A^T + D_{11} B_1^T \end{bmatrix}' P(Q)^{-1} \begin{bmatrix} C_2 \\ C_1 \end{bmatrix} \tag{3.30}$$

is asymptotically stable. This is the discrete time analog of the continuous time filtering equation expressed by $J(\gamma)$ in Theorem 3.1. With this matrix Q, define the

following matrices

$$
\begin{aligned}
Z &:= AQC_1^T + B_1D_{11}^T - (AQC_2^T + B_1D_{21}^T)V^{-1}(C_2QC_1^T + D_{21}D_{11}^T) \\
A_q &:= A + ZR^{-1}C_1 \\
B_{1,q} &:= (AQC_2^T + B_1D_{21}^T)V^{-1/2} + ZR^{-1}(C_1QC_2^T + D_{11}D_{21}^T)V^{-1/2} \\
B_{2,q} &:= B_2 + ZR^{-1}D_{12} \\
C_{0,q} &:= C_0 \\
D_{01,q} &:= (D_{01}D_{21}^T + C_0QC_2^T)V^{-1/2} \\
D_{02,q} &:= D_{02} \\
C_{1,q} &:= R^{-1/2}C_1 \\
D_{11,q} &:= R^{-1/2}(C_1QC_2^T + D_{11}D_{21}^T)V^{-1/2} \\
D_{12,q} &:= R^{-1/2}D_{12}.
\end{aligned}
$$

Finally, define the following matrix functionals:

$$
\begin{aligned}
L(W,Y,K2) &:= \begin{bmatrix} A & B_2 \\ C_1 & D_{12} \end{bmatrix} \begin{bmatrix} Y \\ W \end{bmatrix} Y^{-1} \begin{bmatrix} Y & W^T \end{bmatrix} \begin{bmatrix} A^T & C_1^T \\ B_2^T & D_{12}^T \end{bmatrix} + \\
&\quad \begin{bmatrix} B_1 + B_2K_2 \\ D_{11} + D_{12}K_2 \end{bmatrix} \begin{bmatrix} B_1 + B_2K_2 \\ D_{11} + D_{12}K_2 \end{bmatrix}^T - \begin{bmatrix} Y & 0 \\ 0 & I \end{bmatrix} \qquad (3.31) \\
M_q(Q) &:= C_0QC_0^T + D_{01}D_{01}^T - D_{01,q}D_{01,q}^T \qquad (3.32) \\
M(W,Y,K_2) &:= M_q(Q) + (C_0Y + D_{02}W)Y^{-1}(C_0Y + D_{02}W)^T + \\
&\quad (D_{01} + D_{02}K_2)(D_{01} + D_{02}K_2)^T \qquad (3.33)
\end{aligned}
$$

Let Φ define the set of feasible solutions:

$$
\Phi := \left\{ (W,Y) : Y = Y^T > 0, L(W,Y,K_2) < 0 \right\}, \qquad (3.34)
$$

and consider the optimization problem

$$
\sigma_f := \inf \{ f(M(W,Y,K_2)) : (W,Y,K_2) \in \Phi \}. \qquad (3.35)
$$

Theorem 3.5 *The set of feasible controllers is not empty, if and only if: (1) A stabilizing solution $Q \geq 0$ exists that satisfies the conditions of (3.27-3.30), and (2) $\Phi \neq \emptyset$. If a feasible controller exists, then*

$$
J_f(T_{z_0w}(\mathcal{G}, \mathcal{C})) := f(M(W,Y,K_2)).
$$

Moreover, given any $\alpha > \sigma_f$, there exists a solution , $(W, Y, K_2) \in \Phi$, and the dynamic observer-based controller (with input y and output u)

$$\mathcal{C} := \begin{cases} \sigma x_c = A_q x_c + B_{1,q} r_c + B_{2q} u \\ u = W Y^{-1} x_c + K_2 r_c \\ r_c = Y^{-1/2}(y - C_2 x_c) \end{cases} \tag{3.36}$$

such that $\mathcal{C}$ is feasible and $J_f(\mathcal{G}, \mathcal{C}) = f(M(W, Y, K_2)) < \alpha$ [Ref. 12].

To solve the output feedback generalized $\mathcal{H}_2$ / $\mathcal{H}_\infty$ control problem using Theorem 3.5, the following steps can be followed:

1. Verify that a feasible controller exists; this can be done by solving two standard $\mathcal{H}_\infty$ Riccati equations [Ref. 31].

2. Perform the convex optimization problem: minimize $f(M(W, Y, K_2))$, subject to the constraints $Y > 0$, and $L(W, Y, K_2) < 0$.

3. Construct the output feedback controller $\mathcal{C}$ in (3.36) using the suboptimal solution (W, Y, K_2) obtained in step 2. Then, $\mathcal{C}$ is feasible and $J_f(G, C) < \alpha$; i.e., (3.36) solves the generalized $\mathcal{H}_2$ / $\mathcal{H}_\infty$ control problem.

As with the continuous time problem, where the problem was reduced for state-feedback, the measurement-feedback discrete time problem can be reduced for any problem where the states and disturbances are available for feedback. Consider the following full-information plant:

$$\mathcal{G}_{fi} := \begin{cases} \sigma x &= Ax + B_1 w + B_2 u \\ z_0 &= C_0 x + D_{01} w + D_{02} u \\ z_1 &= C_1 x + D_{11} w + D_{12} u \\ y &= \begin{bmatrix} x \\ w \end{bmatrix}. \end{cases} \tag{3.37}$$

Though the full information structure is not realistic in applications, the solution to this problem gives a lower bound on achievable performance that might yield insight into more complex output feedback problems.

The above optimization problem can be reformulated for the full-information problem by omitting the solution of the Riccati filtering equation (3.27–3.30), omitting the formation of the auxiliary plant, and making the following replacements in the expressions for L and M: A_q is replaced by A, $B_{j,q}$ is replaced by B_j, $C_{i,q}$ is replaced by C_i, and $D_{ij,q}$ replaced by D_{ij}. The optimization functionals are modified by setting $M_q(Q) = 0$. The resulting state-feedback controller is then $\mathcal{C} = K_{sfb} = [WY^{-1}, K_2]$.

The numerical routines which solve both the full-information and measurement feedback controller design problems are outlined in Appendix A. As with the continuous time problem, an ellipsoidal optimization routine was designed to solve the full-information problem. This function file could then either be used to solve a full-information problem directly, or be called by the measurement-feedback design code after the auxiliary plant had been posed. This problem could also have been solved by interior point methods, and the LMI expressions of the matrix functionals can be easily be determined by Schur complements.

C. POLE PLACEMENT BY CONVEX METHODS

Pole placement is possibly the most common metric used to specify the performance of closed-loop systems. The following outlines a convex constraint which can be used to determine state-feedback gains such that a closed-loop system has eigenvalues within a specified circle.

Consider the region D defined by

$$D := \{z \ : \ |z + q| < r, \ q \geq r > 0\}. \tag{3.38}$$

It is a disk in the left half plane with center $(-q, 0)$ and radius r. Let $\alpha := q - r$.

Theorem 3.6 *Given a state-feedback gain matrix $K \in R^{q \times n}$, the eigenvalues of the feedback interconnection of $\mathcal{G}$ and $\mathcal{C} = K$ lie within D if and only if there exists a symmetric matrix $Y > 0$ such that the following Quadratic Matrix Inequality (QMI)*

is satisfied [Ref. 32]:

$$S := (A + B_2K + \alpha I)Y + Y(A + B_2K + \alpha I)^T + (A + B_2K + \alpha I)(Y/r)(A + B_2K + \alpha I)^T < 0. \quad (3.39)$$

By replacing K with WY^{-1}, this expression is equivalent to the following LMI by Schur complements:

$$S_1(W,Y) := \begin{bmatrix} \begin{pmatrix} (A+\alpha I)Y + Y(A+\alpha I)^T + (A+\alpha I)(Y/r)(A+\alpha I)^T + \\ B_2W((A+\alpha I)/r + I)^T + ((A+\alpha I)/r + I)W^TB_2^T \end{pmatrix} & B_2W \\ (B_2W)^T & -Yr \end{bmatrix} < 0. \quad (3.40)$$

The set of state-feedback controllers with eigenvalues within D is therefore convex. As described in [Ref. 33], this constraint is perfectly suited for convex optimal control problems where the designer would like to restrict the search to those controllers whose eigenvalues lie in a specified region.

D. ROBUSTNESS ANALYSIS BY STRUCTURED SINGULAR VALUES (μ)

This section contains a description of the principal tools used to assess the robustness of feedback systems. In particular, we are interested in the use of the structured singular value. The structured singular value (μ) was first introduced by J. Doyle in [Ref. 34] and since has proved to be a valuable tool for the robustness analysis of the closed-loop systems. In this section we briefly summarize the results of [Ref. 34, 35].

Consider the feedback system shown in Figure 3.3. Let $F_l(\mathcal{G}, \mathcal{C})$ denote the feedback interconnection of the plant $\mathcal{G}$ and the controller $\mathcal{C}$ and let $T_{zw}(\mathcal{G}, \mathcal{C})$ denote

the corresponding closed loop transfer function from w to z . It can be shown that

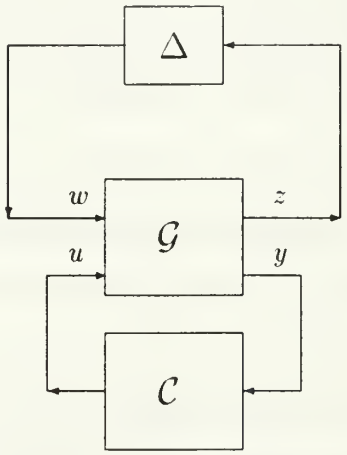

Figure 3.3: Standard feedback configuration with uncertainty block.

any linear interconnection of plant, controller, and uncertainties can be arranged to match the nominal plant, $\mathcal{G}$, having the following form:

$$\mathcal{G} = \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} = \left[\begin{array}{c|cc} A & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array}\right]$$

where (C_2, A, B_2) is stabilizable and detectable. For stability analysis, the controller can be absorbed into the nominal plant $\mathcal{G}$. The LFT (linear fractional transformation) representation of the feedback interconnection of the nominal plant G and the controller $\mathcal{C}$ is given by

$$F_l(\mathcal{G}\ ,\mathcal{C}\ ) = G_{11} + G_{12}\mathcal{C}(I - G_{22}\mathcal{C})^{-1}G_{21}.$$

Recall that the $\mathcal{H}_\infty$ problem does not explicitly address the issue of plant uncertainty. If, however, the plant uncertainty is modeled as an unknown but norm-bounded, stable dynamical system, with the inputs and outputs of the plant uncertainty block included in z and w respectively, then $\mathcal{H}_\infty$ provides stability robustness guarantees as a result of the following theorem:

Theorem 3.7 Small Gain Theorem [Ref. 36]. *Assume $F_l(\mathcal{G},\mathcal{C})$ is stable and Δ belongs to a set*

$$B\Delta := \{\Delta : \Delta \in \mathcal{RH}_\infty, \|\Delta\|_\infty \leq 1/\gamma\}.$$

Then the feedback interconnection of $F_l(\mathcal{G},\mathcal{C})$ and Δ is stable for all $\Delta \in B\Delta$ if and only if

$\|T_{zw}(\mathcal{G},\mathcal{C})\|_\infty < \gamma$.

Unfortunately, Small Gain Theorem can be unnecessarily conservative when applied either to robustness analysis or design. This conservatism can be reduced by using the extension of Small Gain Theorem to structured uncertainties. The structured singular value is the metric by which this extension is applied. In order to define the structured singular value, let

$$\underline{\Delta} = \{diag(\Delta_1, \Delta_2, ..., \Delta_n)\},$$

where Δ_i's are stable FDLTI systems. Furthermore, let

$$\underline{\Delta}_c = \{diag(\Delta_1, \Delta_2, ..., \Delta_n)\},$$

where each Δ_i is a complex matrix.

Definition 3.8 *The structured singular value $\mu(M(j\omega))$ of the complex matrix $M(j\omega)$ is defined as follows:*

$$\mu(M) := \begin{cases} 0, \text{ if } \forall\ \Delta \in \underline{\Delta}_c,\ \det(I + M(j\omega)\Delta) \neq 0 \\ \sup\left\{\|\Delta\|^{-1} : \Delta \in \underline{\Delta}_c, \det(I + M(j\omega)\Delta) = 0,\right\} \text{otherwise.} \end{cases}$$

The importance of the structured singular value for studying robustness of feedback systems is due to the following result [Ref. 34], which characterizes robust stability of a system in the presence of stable structured uncertainty.

Theorem 3.9 *Let*

$$B\Delta = \{\Delta : \Delta \in \underline{\Delta}, \|\Delta\|_\infty \leq 1\}$$

Then the closed loop system of Figure 3.3 is stable $\forall\ \Delta \in B\Delta$ *if and only if*

$$\mu(F_l(\mathcal{G},\mathcal{C}) := \sup_{\omega}(\mu(F_l(\mathcal{G},\mathcal{C}))(j\omega) < 1$$

Applying this theorem as a test for robustness is much less conservative than the small gain theorem itself. By restricting the set of uncertainty matrices to those with the specified structure, a much larger set of LFT's, $F_l(\mathcal{G},\mathcal{C})$, satisfy the stability criteria. In other words, for structured uncertainties, there is a large set of stable systems that fail the criterion of the small gain theorem, and yet satisfy the latter.

IV. THE DESIGN OF AUTOLAND CONTROLLERS FOR CARRIER- BASED F-14 AIRCRAFT BY $\mathcal{H}_\infty$ AND $\mathcal{H}_2$ / $\mathcal{H}_\infty$ METHODS

A. INTRODUCTION

This chapter presents the results of the application of $\mathcal{H}_\infty$ and mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ synthesis techniques to the design of autoland controllers for a carrier based F-14 aircraft. These efforts had several objectives:

- Demonstrate a methodology for $\mathcal{H}_\infty$ output-feedback controller synthesis.
- Expand that methodology for use with $\mathcal{H}_2$ / $\mathcal{H}_\infty$ synthesis tools.
- Demonstrate the use of the $\mathcal{H}_2$ / $\mathcal{H}_\infty$ synthesis tools.
- Demonstrate the feasibility of using Direct Lift in a multi-variable autoland controller.
- Present a methodology for the formulation of uncertainty models based on flight test data.

The nature of the autoland controller problem and the unique configuration of the F-14, made this problem ideally suited to these objectives.

Carrier approach and landing is a challenging multivariable control problem in which the aircraft states must all be carefully controlled in order to comply with multiple structural and safety-of-flight constraints. Automatic landing systems currently in service on carrier-based aircraft (including the F-14) incorporate nested single-input/ single-output (SISO) controllers, which generally seek to regulate the angle-of-attack with the engines in the inner loop, while aerodynamic surfaces such

as elevators or stabilators provide altitude control. Since neither the engines nor the stabilators can control the altitude state directly, their influence is indirect through a combination of other states. The F-14 has an unusual aerodynamic configuration which includes Direct Lift Control (DLC), thereby providing an aero surface with substantial authority to control altitude directly. Moreover, the DLC is driven by actuators whose bandwidth exceeds that of the other control surfaces. Regrettably, this powerful control surface is not used by the automatic landing system currently in service. In this chapter, two multivariable feedback controllers are presented which seeks to exploit this powerful, but dormant capability.

To achieve this objective, the design methodology presented here was developed to enable the control engineer to translate the design requirements into weighting functions for $\mathcal{H}_\infty$ synthesis. This methodology was then further extended for use using the $\mathcal{H}_2$ / $\mathcal{H}_\infty$ design tools. Most typical design requirements are SISO in nature, whereas the $\mathcal{H}_\infty$ and $\mathcal{H}_2$ / $\mathcal{H}_\infty$ synthesis techniques are truly multivariable tools. Thus, the main feature of this methodology is a simple procedure for translating the SISO requirements into the various weighting functions for $\mathcal{H}_\infty$ and $\mathcal{H}_2$ / $\mathcal{H}_\infty$ synthesis. Moreover, once the SISO requirements have been satisfied, the $\mathcal{H}_\infty$ framework offers a natural way to expand the weighting functions to satisfy multivariable stability and performance robustness requirements.

This methodology has been applied to the design of control systems for commercial airplanes, autonomous underwater vehicles (AUV's), flexible structures and, most recently, for unmanned aerial vehicles (UAV's), see [Ref. 6, 7, 8, 9, 10]. In [Ref. 6, 7, 8, 10] this technique was used to synthesize state-feedback controllers. In [Ref. 9], where a controller for a flexible structure was designed, the methodology was extended to include an output feedback case. The methodology outlined here expands this previous work to include compliance with closed-loop sensor bandwidth

requirements.

Specifically, the methodology offers a simple and effective way to design feedback controllers satisfying specified:

- command-loop bandwidths,
- control-loop bandwidths,
- closed-loop damping,
- closed-loop sensor bandwidths.

Each of the above objectives are pursued through a specific formulation of the synthesis model and selection of the various weighting functions. Furthermore, it has been observed that an additional benefit of this methodology is that the resulting controllers do not cancel the undesirable modes of the open-loop plant. This is attributed to the suitable choice of weights to satisfy the closed-loop damping requirement.

The appeal of this methodology is that the control designer is provided with a straightforward framework in which to implement $\mathcal{H}_\infty$ or $\mathcal{H}_2$ / $\mathcal{H}_\infty$ controllers in pursuit of typical design requirements without a detailed understanding of the theoretical basis for these tools. Moreover, the results of the design effort are assessed using familiar SISO figures-of-merit. The availability of good commercial software utilities, and ever-improving computational resources only enhance the viability of iterative design methods such as the one presented here. This methodology is suggested as one means of placing these tools into the hands of practicing control designers.

The controller design methodology we propose necessarily has a heuristic component. This heuristic component has been influenced by our experience in solving the practical problems mentioned above. Although we were successful in applying these design methodologies to this problem, we cannot offer any guarantees as to

whether their application to an arbitrary control problem will yield a satisfactory solution.

Additionally, a method is proposed for accommodating structured model uncertainty in flight dynamics problems. Specifically, in using models extracted from flight test data, the uncertainty in the total value of lift, drag and pitching moment may be less than the uncertainties in individual stability derivatives. For this type of data, a method is proposed that considers the total uncertainties in these forces and moments rather than the uncertainty introduced by each term of the model. The robustness of the resulting closed-loop nonlinear system is then analyzed using established structured singular value methods.

This chapter is organized as follows. Section B contains the material relevant to both controller design problems. This includes a description of the the carrier landing problem and the design requirements. It also includes the details of the uncertainty modeling process which was used to analyze the robustness of the closed-loop systems. Section C presents the details of the $\mathcal{H}_\infty$ controller design process, the implementation of the controller on the nonlinear system, and the analysis of the resulting closed-loop nonlinear system. Section D presents the details of the mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ controller design process, as well as the analysis of the resulting closed-loop system. A comparison of the two design methods and some concluding remarks are then included in sections E and F.

B. PROBLEM STATEMENT

The objective of the controller design is to provide for precise automatic control of the approach and landing of a carrier-based aircraft in the vertical and longitudinal axes. In this section we describe both the plant to be controlled and the desired performance specifications. These specifications are classical in nature, and are rep-

resentative of those posed to control designers in industry. The notational convention adapted in this chapter was to use uppercase letters to denote total values of the variables introduced, while lower case letters denoted the perturbations of these variables around their nominal trim values.

1. Airplane and Model Description

The design problem to be solved here deals with the longitudinal motion of a fighter airplane and the control of the longitudinal rigid body dynamics (by convention, longitudinal flight mechanics refers to the motion in the 2-D plane spanned by the longitudinal and vertical axes). A complete description of an airplane's equations of motion can be found in many available references. See, for example, [Ref. 37]. Consequently, we will not describe the equations of motion in detail. Rather, we will present a brief qualitative description of the key features.

The longitudinal equations of motion of an airplane are described by two force equations (longitudinal force F_x and vertical force F_z) and one angular moment equation (pitching moment M). The state variable associated with the F_x equation is the forward velocity U (along airplane's body-fixed x-direction). The state variable associated with the F_z equation is the angle of attack α (the angle between the body-fixed x-direction and the true total velocity). The state variable associated with the M equation is the pitch rate Q. The integral of Q for a typical approach and landing condition is the pitch attitude Θ (the angle between the body fixed x-direction and the horizon). Other motion variables of interest are the airplane's airspeed V_t (generally not aligned with the body-fixed x-direction), flight path angle γ (the angle between V_t and the horizon), and airplane's altitude above sea level H. All the angles used here are expressed in rad, angle rates in rad/sec, position variables in ft, position rates in fps, and accelerations in g's.

While the above discussion is germane to the flight dynamics of all aircraft, the F-14 has a distinctive aerodynamic configuration due to environment in which it was designed to operate. Mission requirements dictated a variable-sweep wing for low drag at high speeds, with full-span flaps and slats, for very high lift at low carrier take-off and landing speeds. Control along the longitudinal axis is provided by two afterburning turbofan engines ($Thrust$). Pitch control is provided by symmetric deflection of two stabilators ($Stab$). In the landing configuration (gear down, wings fully forward, and flaps fully extended), Direct Lift Control (DLC) is provided by symmetric deflection of wing mounted spoilers. The neutral position of the spoilers is approximately 40% of full deflection, so as to provide for both positive and negative contributions to the lift.

For the design study, the sensors available included onboard accelerometers and gyros which provide pitch attitude (Θ), longitudinal acceleration (N_x), vertical acceleration (N_z), and pitch rate (Q). Total velocity (V_t), was provided by the aircraft's air data system. Lastly, for automated approaches and landings, the altitude (H) was determined by a shipboard tracking radar.

The model used for the design process was a linear model obtained from a nonlinear simulation built using aerodynamic coefficient data. This simulation model was nonlinear in that while the aerodynamic derivatives were held constant, the equations of motion included the nonlinear influence of airspeed, gravity, as well as the dynamic coupling terms. The flight condition was a nominal approach condition of 230 fps, at sea level, with a gross weight of 54,000 lbs. The linearized longitudinal model included five states: u, α, q, θ and h and three control inputs: δ_{Stab}, δ_{Thrust} and δ_{DLC}, where all the linear states and inputs are small perturbations around the nominal operating point. At this condition, the longitudinal rigid body motion of the F-14 is characterized by two second-order stable modes, the phugoid and short

period, and an altitude integrator. The phugoid involves perturbations in V_t and H with nearly constant α, whereas the short period mode involves perturbations in α and Q, with V_t and H remaining constant. The short period mode had a natural frequency of 1.04 rad/sec and a damping ratio of 0.45. The phugoid had a natural frequency of 0.18 rad/sec and a damping ratio of 0.06. In addition to the plants five states, the actuators were modeled by three first-order transfer functions. These were appended to the control inputs and had bandwidths of 20, 2.5 and 50 rad/sec for the stabilators, engine, and DLC, respectively. As a result the complete system was represented by an eighth order linear model.

2. Problem Description

The general problem was to design a feedback controller which would satisfy the operational constraints imposed by the mission. The challenge of landing an aircraft at sea requires very precise control of the aircraft states. Glideslope, which is the desired spatial trajectory of the aircraft, must be tightly controlled to provide for safety and to achieve the precise touchdown necessary to be arrested on the ship. The glideslope is an imaginary ramp oriented three degrees above the horizon, moving with the ship and terminating in the center of the landing area. In the case of manual landings, deviations from glideslope are detected visually by the pilot with a visual reference to a shipboard optical system. For automated landings, a precision tracking radar onboard the ship compares the aircraft's position with an internally calculated glideslope, and transmits an error signal to the aircraft's flight control system via data link. Tight control of aircraft total velocity is driven by the competing requirements of providing for adequate aerodynamic performance, while minimizing the kinetic energy that the airframe and arresting gear must absorb upon landing. Tight control of the aircraft attitude is required to prevent tailstrike. Both of these objectives can be achieved by controlling angle of attack with total velocity and the pitch attitude

as dependent variables, functions of the angle of attack. Angle of attack is also an attractive control variable as consistent aerodynamic performance is achieved for wide ranges of gross weights.

The approach-to-landing problem can be fully characterized by several combinations of the variables in the state vector due to the mathematical and aerodynamic relationships between these variables. Likewise the control objectives can be achieved by tracking any one of these combinations. Systems in fleet use today incorporate nested SISO controllers, with the engine controlling angle of attack in the inner-loop and the stabilators controlling either sink rate or flight path. The remaining variables are then dependent functions of the gross weight and the two controlled parameters. In this design example, we propose to use DLC to provide independent altitude control of F-14 in approach and landing. Currently F-14's DLC is engaged for approach, such that the spoilers are deployed to their neutral DLC position. DLC is not utilized as part of the control system, however, and serves only to increase both the drag and the trimmed power setting (this is done in order to keep the engines in a more responsive range of operation). Neither the engines nor the stabilators provide control directly into the altitude state, but rather indirectly control the flight path/ altitude through the airspeed and pitch attitude states. Performance may thereby be sacrificed, as DLC is the only control effector which has control power directly into altitude through the vertical velocity, and DLC actuator is the fastest of the three available actuators. Given three independent control effectors with sufficient control power, F-14 has the resident capability to track three independent command signals. A multivariable approach to the control design would permit inclusion of the DLC in the control system resulting in both an enhanced capability and an improvement in performance. Therefore our control strategy was to track altitude (H), and angle of attack (α), using stabilators, engines and DLC. Since the number of controllers

exceeds the number of command variables, the flexibility existed to wash out one of the controllers. We chose to wash out the DLC. The desired effect was that the thrust would control the glideslope in steady state, while the DLC would provide dynamic glideslope control. Thus, the design problem was to synthesize feedback controllers which tracked a glideslope signal, while controlling angle of attack in steady-state, given the available sensor suite.

3. Design Requirements

In light of the above, the $\mathcal{H}_\infty$ controller was required to satisfy the following design requirements outlined below. These requirements will later be modified for the mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ design example

1. Zero Steady State Error

- Achieve zero steady state values for all error variables in response to step commands in angle of attack, and ramp commands in altitude (this was necessary for glideslope signal tracking and wind disturbance rejection), while washing out DLC in steady state.

2. Bandwidth Requirements

- The input-output command response bandwidth for all three command channels was to be approximately 1 rad/sec.
- The control loop bandwidth was not to exceed 40 rad/sec for the DLC actuator, 20 rad/sec for symmetric stabilator, and 2 rad/sec for the engine. These numbers represented 80% of the corresponding actuator bandwidths to ensure that the actuators were not driven beyond their linear operating range.

- The sensor response bandwidths were to be approximately 100 rad/sec for the gyros, accelerometers, and integrators, and approximately 5 rad/sec for the altitude, angle of attack and airspeed data.

3. **Closed-Loop Damping**

- The closed-loop eigenvalues associated with physical states were to have the damping ratio of at least 0.6. (This permits controller modes to have damping ratios less than 0.6).

4. **Robustness**

- The controller could not cancel the lightly damped open-loop poles of the plant
- Simultaneous gain and phase margins of $\pm 6dB$ and 45 degrees in all control and sensor loops
- Stability was to be guaranteed for simultaneous variations of 20% in the perturbed lift and drag forces and pitching moment.

4. Uncertainty Modeling

This section describes the methodology used for analyzing robustness of the closed-loop system consisting of the aircraft model and any controller. The aircraft model used in this chapter was obtained from the flight test data. As a result some of the terms in the model are poorly known. These terms include airplane's lift, drag and pitching moment. On the other hand, terms such as gravity and aircraft dynamic coupling are known well. Furthermore, lift and drag are measured in the so-called stability axis, whereas the aircraft model was derived in the body-fixed coordinate system. These considerations indicate that the uncertainties in lift, drag and pitching

moment must be modeled exactly where they occur. Therefore, this should be done using aircraft's nonlinear equations of motion while taking into account the coordinate systems where these forces and moments were measured during the flight test. A detailed discussion of this process is given next.

Let x represent the vector of the longitudinal states of the aircraft:

$$x = [U,\ \alpha,\ Q,\ \Theta,\ H]^T.$$

Let δ represent the vector of the aerodynamic control effectors, consisting of the stabilators and DLC, and let F represent the vector of body-axis forces and moments:

$$\begin{aligned} F_{grav}(x) &:= \text{influence of gravity} \\ F_{dyn}(x) &:= \text{influence of dynamic coupling} \\ F_{aero}(x) &:= \text{influence of aerodynamic forces on the body} \\ F_{thrust} &:= \text{influence of thrust} \\ F_{\delta} &:= \text{influence of aerodynamic forces on the control surfaces.} \end{aligned}$$

Note that the first three are state dependent, while the last two are functions of the appropriate controllers. The aircraft equations of motion can now be expressed as:

$$\begin{bmatrix} \dot{U} \\ \dot{\alpha} \\ \dot{Q} \end{bmatrix} = F_{grav}(x) + F_{dyn}(x) + F_{thrust} + R_{wb}(x)\,(F_{aero}(x) + F(\delta)), \tag{4.1}$$

where $R_{wb}(x)$ is the wind to body axis rotation matrix:

$$R_{wb} = \begin{bmatrix} -\cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & -\cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.2}$$

We only consider uncertainties in the aerodynamic forces and moments, since the gravity, dynamic coupling and thrust are all well known.

The aerodynamic model of the airplane is derived using stability derivatives, which represent the contribution of each state and control input to the aerodynamic forces and moments acting on the airplane. This data can be obtained either in the wind tunnel, where isolation of individual contributions is frequently possible, or flight test, where only macroscopic behavior is observed, and then numerically distributed among the aircraft's states and inputs. The stability derivatives obtained from the flight test data therefore depend upon very complex multivariable parameter identification (PID) methods, which are executed in two steps. First, forces and moments are computed from observed accelerations and rates. Second, the PID process tries to identify which control inputs and airplane states contribute to the observed change in forces and moments. This step clearly introduces errors not present in the first computation.

Our choice is then to consider either the contribution of the uncertainty in each parameter, or the net uncertainty in our knowledge of the forces and moments. There are several reasons to choose the latter. First of all, the size of the uncertainty model is significantly reduced. Secondly, the uncertainties in each stability derivative are not independent. If they were, then large uncertainties in each stability derivative will result in large uncertainties in the net forces or moments, leading to an unnecessarily conservative design. Consequently, we chose to model the uncertainty block, Δ, as a 3×3 diagonal matrix, where each diagonal element represented a percentage of the nominal perturbation in the aerodynamic forces and moments (drag, lift and pitching moment). Incorporating this block in equation 4.1 yields:

$$\begin{bmatrix} \dot{U} \\ \dot{\alpha} \\ \dot{Q} \end{bmatrix} = F_{grav}(x) + F_{dyn}(x) + F_{thrust} + R_{wb}(x)\,(I + \Delta)\,(F_{aero}(x) + F(\delta))\,. \qquad (4.3)$$

Figure 4.1 depicts equation 4.3. Here signals w_δ and z_δ denote the uncertainty inputs and outputs.

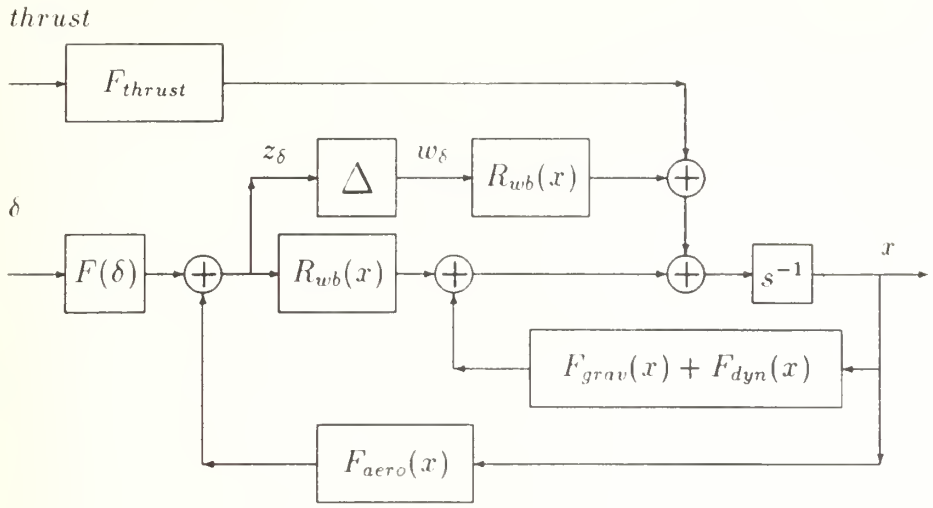

Figure 4.1: Uncertainty Model

C. $\mathcal{H}_\infty$ CONTROLLER DESIGN

In this section, we will describe the key features of the controller design process which we followed. The section is organized into a number of subsections that emphasize some of the important engineering issues that arose in the controller design. The MATLAB scripts and SIMULINK models used to pose and analyze the problem are outlined in Appendix C.

1. Synthesis Model

The first step in the controller design process was the development of the synthesis model which served as an interface between the designer and the $\mathcal{H}_\infty$ controller synthesis algorithm.

Consider the feedback system in Figure 4.2. The synthesis model was derived from the linear model of the airplane by appending the depicted weights . The weights became the "knobs" which the designer adjusted to achieve his specified performance. Here $\mathcal{C}$ is the controller to be designed, P is the linear model of the F-14 and the block $\mathcal{G}$ within the dotted line is the synthesis model. The signal w_1

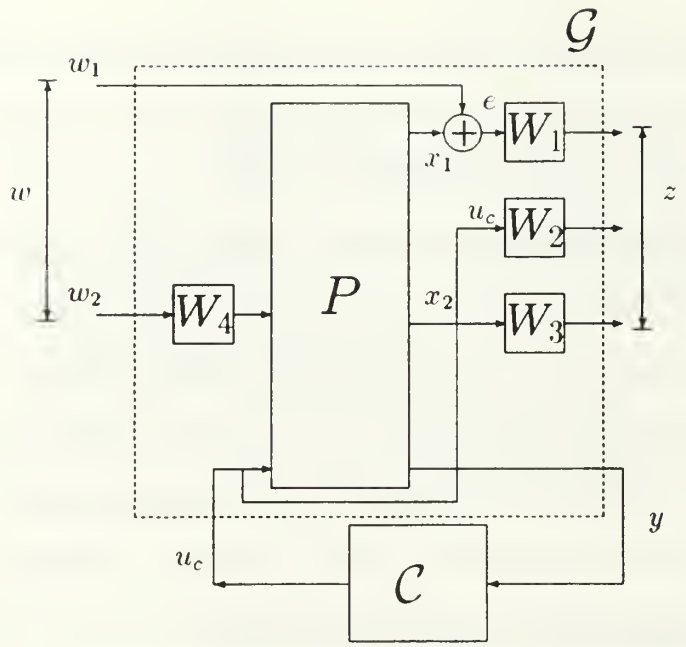

Figure 4.2: Synthesis Model

represents the commanded inputs which were to be tracked:

$$w_1 = \begin{bmatrix} h_{cmd} & v_{cmd} & \theta_{cmd} \end{bmatrix}'$$

The signal w_2 represented the noise inputs to each of the sensors, and disturbance inputs to the states of the plant. The signal u_c represented the control inputs to the system and was composed of the stabilator command, the thrust command, and the DLC command. The signals x_1 and x_2 are:

$$x_1 = \begin{pmatrix} h & \alpha & DLC \end{pmatrix}' \qquad x_2 = \begin{pmatrix} \dot{u} & \dot{\alpha} & \dot{q} & \dot{\theta} & \dot{h} \end{pmatrix}'.$$

The signal e represented the vector of the tracking errors ($e = w_1 - x_1$).

The outputs of W_1, W_2 and W_3 comprised the vector z. Since we required zero steady-state errors in tracking a ramp altitude command, and a step α and DLC commands, the weighting function W_1 was chosen to have the following form:

$$W_1 = \begin{pmatrix} \frac{c_1}{s^2} & 0 & 0 \\ 0 & \frac{c_2}{s} & 0 \\ 0 & 0 & \frac{c_3}{s} \end{pmatrix}.$$

where the constants c_1, c_2 and c_3 were adjusted to get the desired command response bandwidths. Thus, W_1 weighs the integrators on the regulated variable error channels. Furthermore, it was required to have full rank in order to satisfy the detectability assumption of Theorem 3.1.

It turned out, that unlike W_1, W_2 and W_3 did not need to include any dynamics. The choice for the weighting function W_2 was:

$$W_2 = \begin{pmatrix} c_4 & 0 & 0 \\ 0 & c_5 & 0 \\ 0 & 0 & c_6 \end{pmatrix},$$

where c_4, c_5, and c_6 were adjusted to achieve the desired control loop bandwidths; W_2 was also required have full rank in order to satisfy the full rank assumption of Theorem 3.1.

Next, the reader will note that the elements of the vector x_2 are the rate terms on the principal states of the plant. Selection of x_2 is an important element of our methodology. Applying the weight W_3 to x_2, and including these signals in z, penalized activity in their corresponding states. The effect was similar to that of creating rate feedback to augment damping, common to a classical controls approach. Importantly, W_3 did not need to have full rank. This permitted us to set multiple weights to zero and use non-zero values only in the event that a particular signal was necessary to improve damping. As a result W_3 had the following form:

$$W_3 = \text{diag}(c_i), \quad i = 7, \ldots, 11,$$

where c_i's were used to improve damping in a particular mode, as will be discussed in the next section.

The vector y included the system's sensor outputs. Furthermore, y had to include the integral error state in order to satisfy the (C_2, A, B_2) detectability

assumption of Theorem 3.1. Consequently, y was comprised of:

$$y = \left[h \;\; \alpha \;\; v_t \;\; \theta \;\; n_z \;\; n_x \;\; q \;\; \frac{h_e}{s^2} \;\; \frac{\alpha}{s} \;\; \frac{DLC}{s} \right]'.$$

One artificiality was introduced at this point: the presence of the integrated error terms in y necessitated the inclusion of noise signals on those measurements within the synthesis model in order to satisfy the rank condition on D_2. Since the bandwidth of these "observations" were arbitrary, we set them to the highest frequency of the other observations.

In summary, the cost function penalized a vector of the weighted integrated errors, the rates on principal plant dynamic states and the control inputs. The design process to be discussed is essentially a procedure for adjusting the weights on W_1, W_2 and W_3 in order to achieve the design specifications.

2. The Design Procedure

The design process is summarized next, followed by a detailed discussion of how each of the design steps were applied to the design example:

1. Set all W_3 weights to zero. Use state-feedback design to determine weights for W_1 and W_2 to satisfy the command and control-loop bandwidth requirements.

2. If damping was unsatisfactory, refine the state-feedback design by adjusting W_3 weights to include lightly damped states in output z. These states were identified by examining the eigenvectors associated with lightly damped eigenvalues of $A + B_2 K_{sfb}$, where K_{sfb} was the state-feedback gain determined in step 1 above. The maximum element of that eigenvector corresponded to the state contributing most to the lightly damped mode. Increasing the weight in the corresponding W_3 entry had the effect of damping the dynamic activity of that state. Readjust W_1 and W_2 weights to maintain the previously achieved bandwidth specifications.

3. Given $W_{1,2,3}$ weights determined above, use measurement feedback design to determine the sensor noise weights in W_4 necessary to satisfy sensor response bandwidths.

4. Determine the process noise weights in W_4 by analysis of the broken-loop controller responses, adjusting the weights as necessary to match the cross-over frequencies that were observed for the state-feedback design. This step is similar to Loop Transfer Recovery (LTR) technique developed for linear quadratic methods.

5. Readjust $W_{1,2,3}$ as required to maintain previously achieved specifications.

6. Evaluate resultant controller using linear and nonlinear simulation. Adjust weights as necessary.

7. Confirm satisfaction of other specification elements: robustness, damping, and no cancellation of lightly damped open-loop poles.

a. State-Feedback Design - Determining the W_1 and W_2 Weights

The objective of this step was the determination of the appropriate W_1 and W_2 weights to achieve the specified command and controller bandwidths. Visual inspection of bode plots for the broken-loop controller responses and the closed-loop command responses indicated which weight to adjust for the next design iteration. As a general rule, increasing the weight on the integral errors (W_1) increased the bandwidth of the respective command response channel, and increasing the weight on the controller commands (W_2) decreased the broken-loop cross-over frequency. This is identical to the behavior noted in $\mathcal{H}_2$ design, and is consistent with intuition, i.e., increasing the relative cost of the integral error should increase the closed-loop responsiveness of that channel in order to rebalance the costs. Similarly, increasing

the relative cost on a specified control signal should decrease the amount of energy applied to that control channel, and result in a decrease of that controller's bandwidth. While adjustment of the weight on a given term consistently had the desired effect on the corresponding bandwidth, internal coupling meant that the influence of a given weight was not unidirectional and occasionally resulted in wild variations in the other Bode traces, invariably in the most undesirable direction. Occasionally, significant variations could also be attributed to the binary search stopping on a value much closer to the optimal than the tolerance. Considerable time could be expended chasing the various Bode traces, if an orderly methodology was not followed. Generally, once a desired cross-over or corner frequency was attained, weights were then adjusted in response to adverse coupling effects to restore that value to spec before any further adjustment of other bandwidths. Table 4.1 depicts both the nominal performance and the refined performance for several iterations as the weights were varied. Each entry represents the weight applied to the identified term in z in the numerator, and the resulting bandwidth in that channel in the denominator. The column for damping ratio represents the minimum damping ratio for all complex closed-loop poles. The frustration attendant with coupling across terms is evident from iteration two to three. Raising the second weight from the first to second iteration had successfully pushed the α response up to above one, but had an undesirable influence on the DLC response.

b. Improving the State-Feedback Design: Rate Feedback for Damping

By the third iteration, the bandwidths had been balanced close enough to the desired specifications to attempt improvement of closed-loop damping. The objective here was to identify the principal states participating in under-damped modes, and increase the weight on their respective rates in the output z such that activity

TABLE 4.1: STATE-FEEDBACK DESIGN: WEIGHTS/RESULTING BANDWIDTHS

iter.	W_1			W_2			W_3					ζ
	h_e	α_e	DLC	$\delta_{stab_{cmd}}$	$\delta_{thrust_{cmd}}$	$\delta_{DLC_{cmd}}$	$\dot{u}$	$\dot{w}$	$\dot{q}$	$\dot{h}$	w	
spec	$> 1s^{-1}$	$> 1s^{-1}$	$> 1s^{-1}$	$< 20s^{-1}$	$< 2s^{-1}$	$< 40s^{-1}$	-	-	-	-	-	> 0.6
1	1/ 1.5	1/ .01	1/ 1	1/ 2.5	1/ .01	1/ 1	0	0	0	0	0	0.43
2	1/ .8	10/ 2	1/ .01	1/ 4.5	1/ .01	1/ 1	0	0	0	0	0	0.43
3	1/ .8	5/ 1	1/ 1	1/ 4	1/ .01	1/ 1	0	0	0	0	0	0.45
5	1/ .9	5/ 1	1/ 1	1/ 100	1/ .01	1/ 1	0	0	5	0	0	0.53
7	1/ .9	5/ 1	1/ 1	1/ 100	1/ .01	1/ 1	0	5	5	0	0	0.62
8	1/ .8	5/ 1	1/ 1	5/ 8	1/.01	1/ 1	0	5	5	0	0	0.44
24	10/ 1	30/ 3	5/ 20	5/ 10	.01/2	1/ 6	0	10	5	1	0	0.60

in that mode was penalized. The intent was identical in philosophy to the purpose of rate feedback in classical control design. At the third iteration, the closed-loop system matrix $(A + B_2 K_{sfb})$ had two complex pairs of eigenvalues with damping ratios of 0.55 and 0.43. The two eigenvectors corresponding to these under-damped modes pointed in the directions of the h/s and q states respectively. This indicated that h/s and q were the dominant participants in each of the two under-damped modes. Since h/s was a state internal to the controller it was disregarded, and attention was focused on enhancing the damping on q. To improve the damping of these modes, the weighted output $\dot{q}$ was included in z and the value of the corresponding term in W_2 was increased to a non-zero value.

Initially a very small weight was introduced relative to the other weights. Each time a damping weight was adjusted, the W_1 and W_2 weights were readjusted in response, to regain the bandwidth characteristics previously achieved, prior to further refinement of the damping weights. The eigen analysis was performed prior to each adjustment of the damping weights to identify the most active state of each under-damped mode, and to insure that a different mode with different modal participation had not become under-damped. In fact, by penalizing participation of q in the under-damped mode, the mode shifted into the α and θ states, requiring the

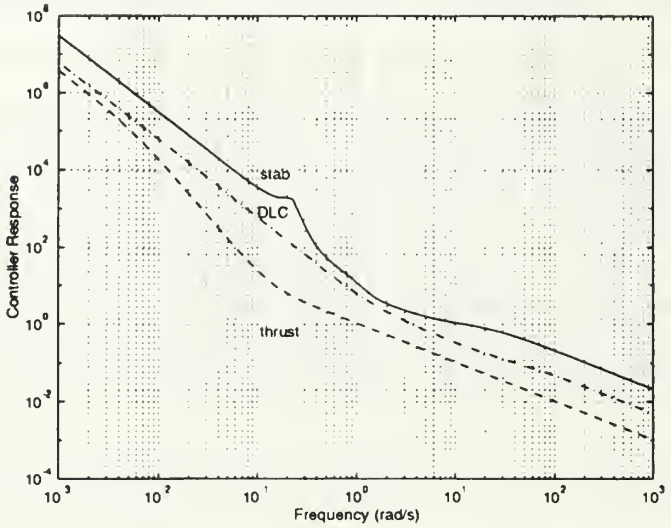

Figure 4.3: Broken Loop Controller Responses

inclusion of a penalty against the respective rate terms. Rarely was more than one weight adjusted per design iteration, to preclude obscuring the influence of each adjustment. A total of 24 iterations were required to achieve the design that approached both bandwidth and damping specifications. From Table 4.1, the weighting functions W_1, W_2 and W_3 at the conclusion of this phase had the following values:

$$W_1 = \begin{pmatrix} \frac{10}{s^2} & 0 & 0 \\ 0 & \frac{30}{s} & 0 \\ 0 & 0 & \frac{5}{s} \end{pmatrix}, \quad W_2 = \begin{pmatrix} 5 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad W_3 = \text{diag}(0, 10, 5, 1, 0).$$

Figures 4.3 and 4.4 depict the broken-loop controller responses, and the closed-loop command responses for this selection of weights.

c. Output Feedback Controller Design - Selecting the Measurement Noise Weights

The next step was to determine the weights to be applied to the measurement noise signals, which mathematically show up only in the D_2 matrix of the synthesis model. Initially, the weights on the regulated output z determined by the state-feedback design process were used in the output feedback design. The

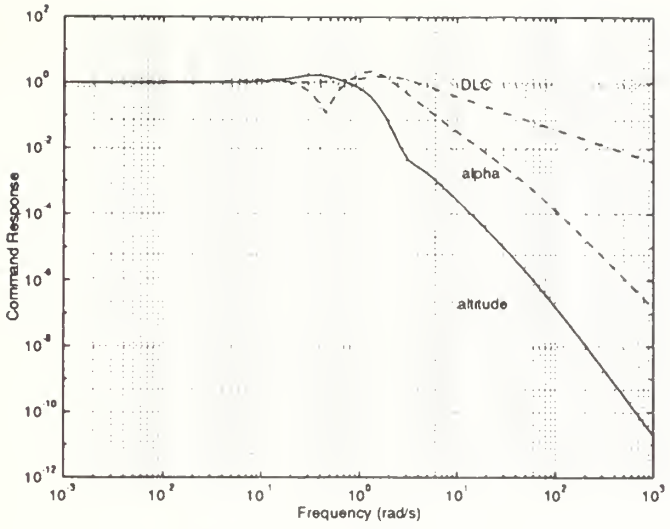

Figure 4.4: Closed-Loop Command Responses

objective of this phase was to tune the controller such that the influence of each sensor channel on the controller did not exceed the expected reliability of the sensor. For example, if a sensor could be regarded as reliable at frequencies up to 10 rad/s, then the controller response to that sensor channel should roll off at a frequency less than or equal to 10 rad/s. Consider the following representation of the controller:

$$\mathcal{C} = \begin{cases} \dot{\zeta} &= A_c\zeta + B_c y \\ u &= C_c\zeta \end{cases} \tag{4.4}$$

The frequency responses of the diagonal terms of the transfer function matrix $C_2(sI - A_c)^{-1}B_c$ were plotted to evaluate the influence of each channel on the controller (Figure 4.5). The outputs of this transfer matrix represent the estimates of the states of $\mathcal{G}$ in the presence of the worst case disturbance [Ref. 18]. By examination of the corner frequency for each of the ten channels of $C_2(sI - A_c)^{-1}B_c$, the weights could be adjusted to achieve the desired sensor response bandwidth. Increasing the weight on a given term resulted in the bandwidth for that channel being decreased. This is similar to the results encountered in $\mathcal{H}_2$ design and is consistent with intu-

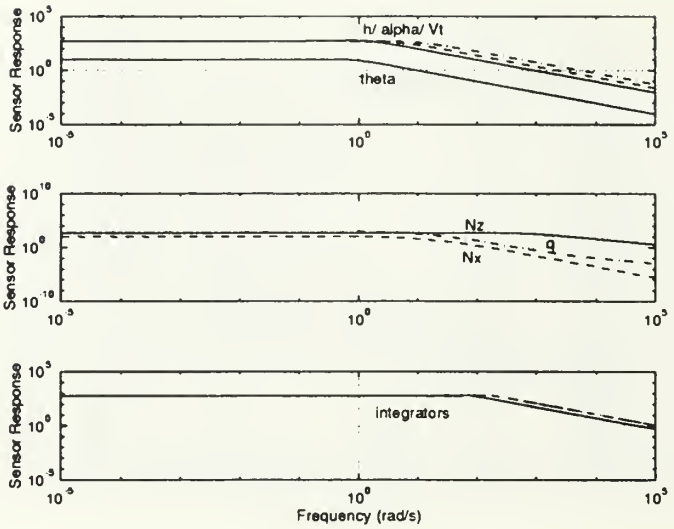

Figure 4.5: Sensor Responses

ition. Very little effort was required to find the appropriate set of weights since little coupling was observed between the sensor channels. The accelerometer bandwidths could not be exceed 10 *rad/sec* because of the internal dynamics of the controller.

d. Output Feedback Controller Design - Selecting the Process Noise Weights

The next step of the design process was the determination of the process noise weights. This phase was similar in execution to the "loop recovery" process of LTR, with a slightly different purpose. While the objective with LQG/LTR is principally the recovery of state-feedback robustness properties, the objective here was the recovery of the performance characteristics of the state-feedback controller, as reflected by the various bandwidth and damping specifications. Unlike each of the previous steps, in which a specific weight could be expected to control a particular trace on the graphs, this was not so for this phase. Additionally, there was coupling into the closed-loop sensor responses as the state process noise weights were changed, which required some additional adjustment of the sensor noise weights to maintain

the desired sensor response bandwidths. The final value of W_4 was:

$$W_4 = 0.00001 * \text{diag}(0.1,\ 4,\ 4,\ 0.01,\ .01,\ .01,\ 0.1,\ 10,\ 10,\ 1,\ 5,\ 5,\ 5\ )$$

e. Linear Simulation- Assessing the Controller Structure

At this point the closed-loop linear system was simulated in order to determine whether reasonable actuator deflections were being used, and to ensure that an altitude ramp could be tracked while controlling α to a desired trim value and DLC to zero. The closed-loop system was initialized to level flight and then expected to intercept and track an altitude ramp. While the altitude ramp was successfully intercepted and tracked, both α and DLC stabilized at values other than their respective set points. Examination of the transfer functions from altitude command to α and DLC revealed only a single zero at the origin within each numerator. In both cases there was an additional zero numerically close to zero, but insufficient to provide the desired washout characteristics. To achieve these characteristics, an additional integrator was added to both channels in W_1. A new controller was then obtained using this modified synthesis model with the same scalar weights determined above. This controller was then evaluated using the identical simulation, with the result that both α and DLC stabilized at their desired values. Finally, slight change in W_1 and W_2 resulted in the final determination of the bandwidths. The final values of the weighting matrices were:

$$W_1 = \begin{pmatrix} \frac{10}{s^2} & 0 & 0 \\ 0 & \frac{20}{s^2} & 0 \\ 0 & 0 & \frac{6}{s^2} \end{pmatrix},\quad W_2 = \begin{pmatrix} 5 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 1 \end{pmatrix},\quad W_3 = \text{diag}(0, 10, 5, 1, 0).$$

Figures 4.6 and 4.7 depict the resulting broken-loop controller responses and the closed-loop command responses. The Nyquist plots of the broken loop responses for each control input are shown in Figure 4.8.

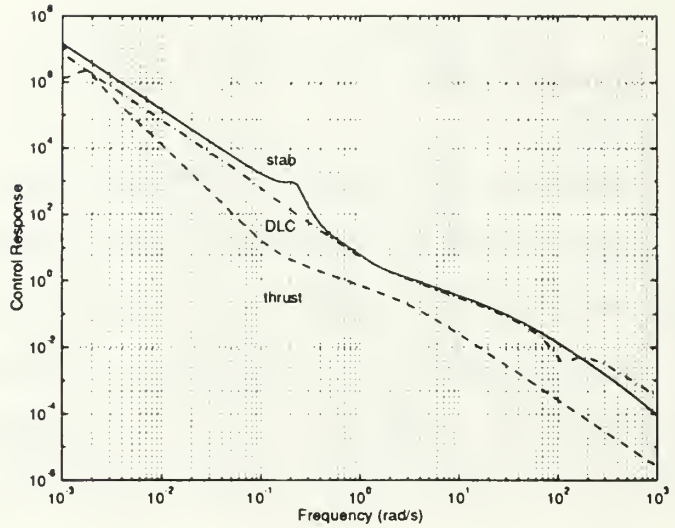

Figure 4.6: Output Feedback Broken-Loop Controller Responses

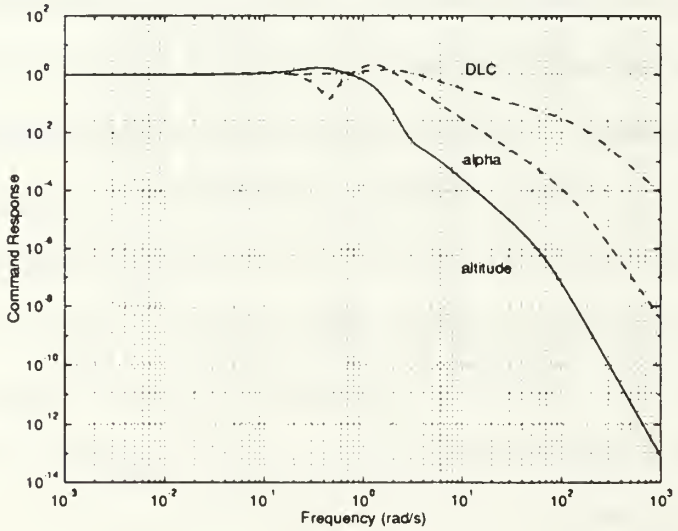

Figure 4.7: Output Feedback Closed-Loop Command Responses

3. Specification Compliance

Figures 4.6 and 4.7 show all the control and command loop bandwidths as satisfying their respective specifications, with the exception of the α command response which is slightly low at approximately 0.8 rad/sec. The notch is due to the

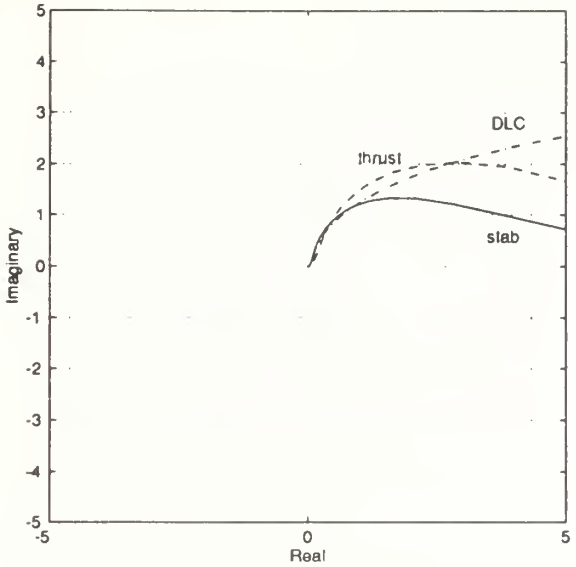

Figure 4.8: Output Feedback Broken-Loop Nyquist Plot

presence of a zero in the α command loop. No objectionable properties result since that channel is used to regulate a constant command rather than respond to changes in the command signal. The Nyquist plot in Figure 4.8 clearly depicts the simultaneous phase and gain margins requirements as being satisfied, with all three controller traces staying in the right half plane for all frequencies. Eigen decomposition of the closed-loop system revealed a complex pair of eigenvalues which failed the damping ratio requirement (0.45). Examination of the corresponding eigenvectors, however, revealed that no vehicle states and only controller states were participating in this mode.

The next step was to ensure that the cancellation of the plant's lightly damped modes by the controller had not occurred. The controller is a 3×9 matrix of transfer functions, and while cancellation may occur in one or several channels, complete cancellation would only occur if all the channels had a common numerator term. See [Ref. 38] for a discussion of multivariable transmission zeros. Figure 4.9 depicts the singular values of the open-loop control responses. The presence of a clear spike at the frequency of the lightly damped phugoid mode and a small bump at the

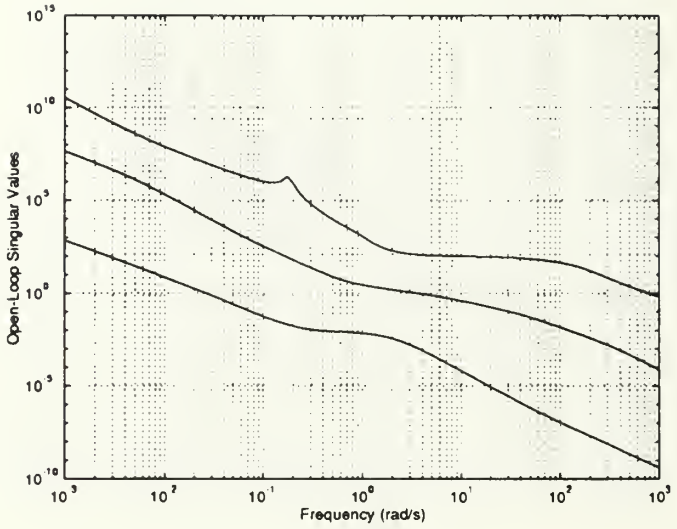

Figure 4.9: Control Loop Gain Singular Values

frequency of moderately damped short period mode indicates that the controller has not canceled these open-loop poles. Numerical analysis revealed no transmission zeros in the controller, further confirming the absence of pole-zero cancellations.

4. μ Analysis

The robustness analysis was performed by determining the structured singular value of a linearization of the closed-loop uncertainty model consisting of the nonlinear plant and the $\mathcal{H}_\infty$ controller, as discussed previously in section B.4. This linearization was performed about the trimmed operating condition, with the uncertainty inputs and outputs w_δ and z_δ included in the nonlinear equations of motion as shown in Figure 4.1. Figure 4.10 depicts the resulting structured singular value, where the peak value of 0.6 indicates compliance with the condition of Theorem 3.8 and the robustness design specification.

Note, had the design effort failed to yield the desired robustness, the design process could have been repeated with the signals w_δ and z_δ incorporated in the synthesis model, and using either $\mathcal{H}_\infty$ synthesis or D-K iteration [Ref. 35].

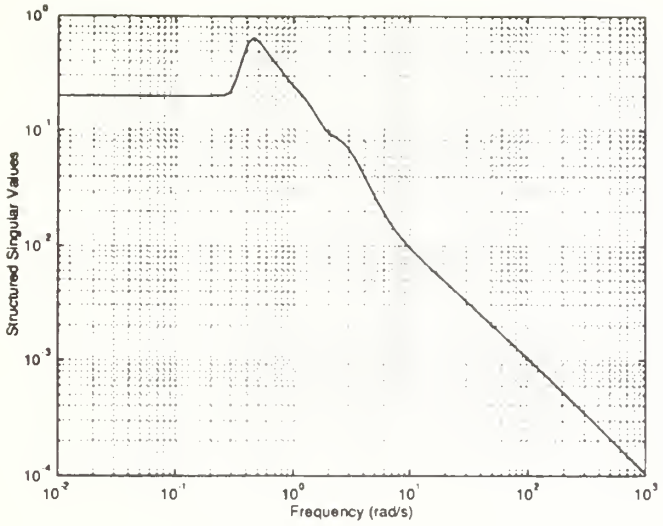

Figure 4.10: Structured Singular Value Plot

5. Nonlinear Simulation

In this section we present the results of the nonlinear simulation of the $\mathcal{H}_\infty$ controller. First, the controller was implemented on the nonlinear plant using $\mathcal{D}$-implementation methodology (see [Ref. 39]). The methodology is based on the observation that linear controllers are designed to act on the perturbations about the plant's nominal trajectory. Next, as with the linear simulation performed in section C.2.e, the controller task was to intercept and track an altitude ramp, while appropriately controlling the other signals of interest. A vertical and horizontal gust field of moderate intensity (rms= $10fps$) was included in the simulation. Results are depicted in Figure 4.11, where all the variables are shown as deviations from the trimmed level flight condition. The simulation was initialized in steady level flight with all surfaces at their trim positions. In response to the altitude ramp, the DLC and stabilizer deflected immediately to establish the appropriate descent rate. The result is a decrease in both α and Θ. The altitude overshoots the ramp, and then corrects to the proper trajectory. Both the average angle of attack and the average

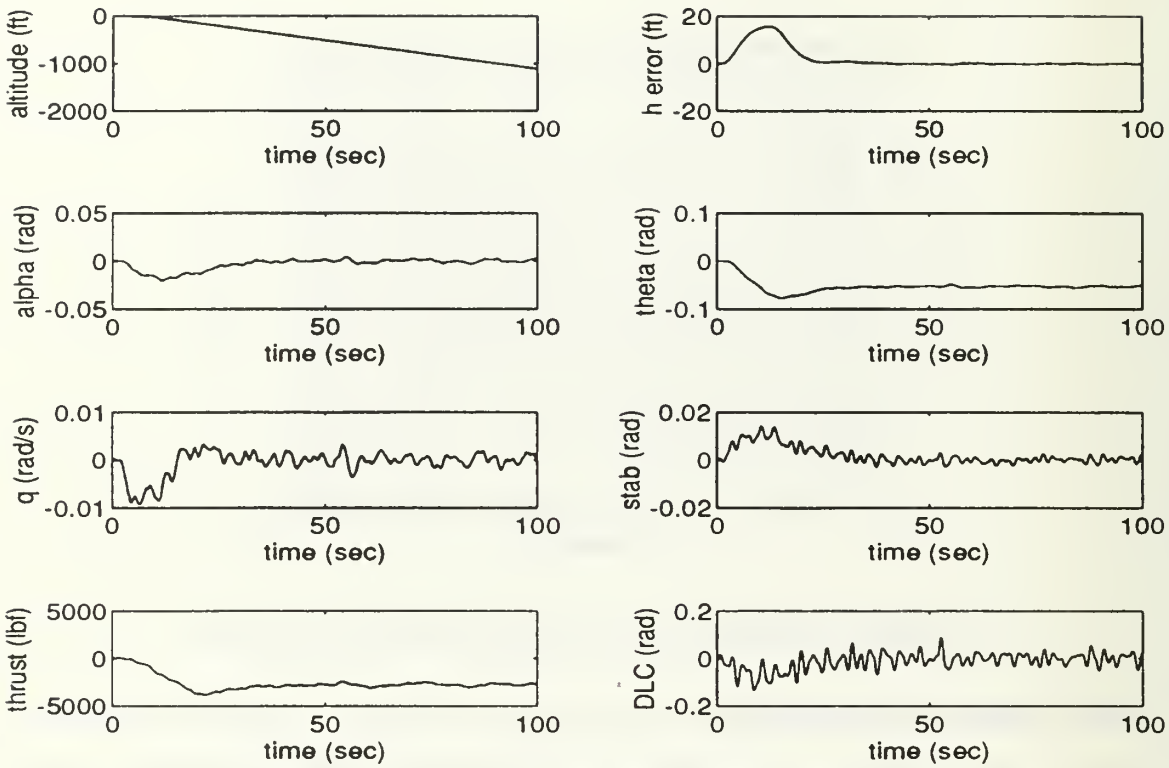

Figure 4.11: Nonlinear Simulation Results

DLC wash-out to their desired trim values as the altitude error is nulled and the thrust decreases to stabilize at the new steady-state condition. Deflections and deviations are all well within reasonable practical values. This simulation validated the results of the design effort.

6. $\mathcal{H}_\infty$ Design Conclusion

A measurement feedback controller was successfully designed to provide longitudinal control of an F-14 aircraft during automatic landing, and implemented on a nonlinear simulation. A key feature in the design was the exploitation of the aircraft's Direct Lift Control to provide for enhanced landing performance. Additionally, a methodology was detailed whereby SISO performance requirements were achieved using $\mathcal{H}_\infty$ synthesis. Finally the resulting controller was validated on the nonlinear simulation.

D. MIXED $\mathcal{H}_2$ / $\mathcal{H}_\infty$ CONTROLLER DESIGN

The purpose of this section is to present the design example demonstrating both an application of the $\mathcal{H}_2$ / $\mathcal{H}_\infty$ design tools, and a methodology for their use. The previous problem was actually born out of a desire to fully develop the methodology for the pure $\mathcal{H}_\infty$ problem prior to its application to a more complicated mixed problem. Both the synthesis and analysis models used the identical MATLAB scripts and SIMULINK models as the above problem. The design tools for this problem were created by the author and are developed and outlined in Appendix A.

1. Problem Description and Design Requirements

The problem description and design requirements for this example were identical to those outlined in section B above, with slight modification. In order to demonstrate the attributes of the mixed design tools, a more severe robustness requirement was included. Specifically, stability was to be guaranteed for simultaneous variations of 40% in the perturbed lift and drag forces and pitching moment. In the previous example, the robustness of the closed-loop system was assessed after the design process. In this design example, it was decided to explicitly impose the robustness requirement in the design process using the $\mathcal{H}_\infty$ feature of the design tools. To accommodate the increased robustness specification, it was decided to relax the desired performance requirement slightly. Rather than the command performance bandwidths of approximately one rad/sec which had previously been required, the desired command bandwidths were set at approximately 0.5 rad/sec.

2. Synthesis Model

The $\mathcal{H}_\infty$ design tools used in the first design example found the controller, $\mathcal{C}$, that minimized $\|T_{zw}(\mathcal{G}\ ,\mathcal{C}\ )\|_\infty$. The mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ tools find the controller that minimizes the mixed cost: $\|T_{z_0w}(\mathcal{G}\ ,\mathcal{C}\ )\|_{2/\infty}$, subject to the constraint

$\|T_{z_1 w}(\mathcal{G}, \mathcal{C})\|_\infty < \gamma$. Consequently, the synthesis model complexity was increased in order to accommodate the signal z_0, which was not present in the previous example. From Chapter III, the mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ problem required that the synthesis model be posed in the form:

$$\mathcal{G} := \begin{cases} \dot{x} &= Ax + B_1 w + B_2 u \\ z_0 &= C_0 x + D_{01} w + D_{02} u \\ z_1 &= C_1 x + D_{11} w + D_{12} u \\ y &= C_2 x + D_{21} w + D_{22} u \end{cases} \tag{4.5}$$

In order to pose the problem in this form, consider Figure 4.12. This figure depicts the synthesis model where the signals were chosen to comprise each input and output vector. The choice of signals to include in w, z_0 and z_1, as well as the various weighting functions, were determined both by mathematical necessity, and the objectives posed by the problem description.

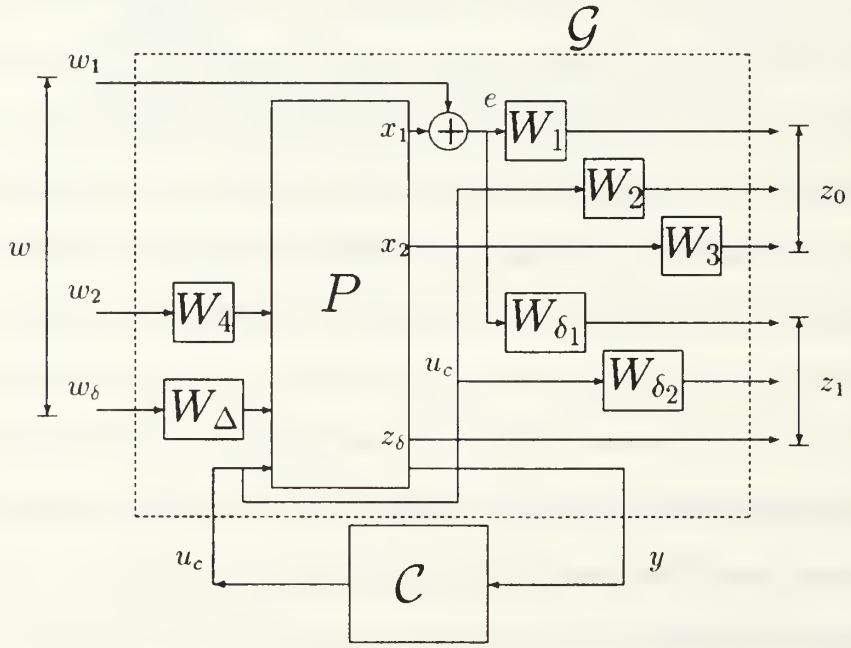

Figure 4.12: Mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ Synthesis Model

The vector z_1 consisted of the signals whose output energy would be limited by the $\mathcal{H}_\infty$ constraint. Since the $\mathcal{H}_\infty$ constraint was to be used to explicitly

achieve the specified robustness, the uncertainty input w_δ was included in w, and the output of the uncertainty model z_δ, was included in z_1. The uncertainty model depicted in Figure 4.1 was used to find a linearization of the open-loop system with the input vector w_δ, and output vector, z_δ. As the consequence of constraining $\|T_{z_1 w}(\mathcal{G}, \mathcal{C})\|_\infty < \gamma$, $\|T_{z_\delta w_\delta}(\mathcal{G}, \mathcal{C})\|_\infty < \gamma$ would be guaranteed. The requirement that $\begin{bmatrix} A \\ C_1 \end{bmatrix}$ have full column rank necessitated the inclusion of the output of any pure integrators, and the requirement that D_{12} have full column rank necessitated the inclusion of the control inputs u_c in z_1 as well. In order to restrict the amount of conservatism that these two signals added to the $\mathcal{H}_\infty$ constraint by their presence in z_1, small attenuating values were chosen for the weighting functions W_{δ_1} and W_{δ_2}. After several iterations of the design process it was evident that double integrators would again be required on each of the regulated error channels. Consequently, the final weighting matrices on z_1 had the form:

$$W_{\delta_1} = \frac{10^{-3}}{s^2} I_3 \qquad W_{\delta_2} = \mathrm{diag}(10^{-3}, 10^{-6}, 10^{-3})$$

The second element of W_{δ_2} was specifically smaller than the others because it multiplied the thrust channel, whose units (lbf) resulted in high signal amplitudes. The weighting function W_Δ was set to $0.4I$ in order to scale the uncertainty model such that $\|T_{z_\delta w_\delta}(\mathcal{G}, \mathcal{C})\|_\infty < \|T_{z_1 w}(\mathcal{G}, \mathcal{C})\|_\infty < \gamma = 1$ would satisfy the robustness specification of 40% gross parametric variation. Each of these weighting functions were constants and not adjusted during the design procedure.

The vector z_0 represented those signals which would appear in the quadratic cost function. As with the previous pure $\mathcal{H}_\infty$ problem, the weighting functions W_1, W_2, W_3, and W_4 consequently represented the degrees of design freedom, while constrained by the above $\mathcal{H}_\infty$ specification. In order to achieve similar results as the previous example, z_0 was selected to be identical to the previous z, as was the

structure of the weighting matrices. To be able to adjust the command bandwidths, the integrated errors were again included in the output signal, resulting in the first weighting matrix having form:

$$W_1 = \begin{pmatrix} \frac{c_1}{s^2} & 0 & 0 \\ 0 & \frac{c_2}{s^2} & 0 \\ 0 & 0 & \frac{c_3}{s^2} \end{pmatrix}.$$

As in the previous problem, W_2, W_3, and W_4 were diagonal matrices of constants.

3. The Design Procedure

The design process was virtually identical to that outlined in section 2. above. The only exceptions were in the formulation of the synthesis model, as described above, and in the use of the mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ design functions outlined in Appendix A, in lieu of the commercial $\mathcal{H}_\infty$ design tools. Due to lack of time, the design example was terminated with the design of a state-feedback controller, however, Appendix A does include both the state-feedback and measurement feedback design routines.

Since the procedure otherwise followed the procedure of the first example exactly, the details are omitted. The mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ design tools did permit the selection of which generalized mixed costs was to be used (trace, maximum eigenvalue or maximum diagonal element). Several iterations were done using both the trace and the maximum eigenvalue. It was observed that the trace was more appropriate to this particular methodology, in that it was easier to influence all of the various bandwidths by adjustments of the weighting matrices. Several bandwidths were comparatively insensitive to adjustments of the weights when using the maximum eigenvalue as the cost functional. This is consistent with intuition when one considers the geometric implications of choosing the trace relative to the maximum eigenvalue. Table 4.2 summarizes the progress of the design effort using the trace. Each design iteration required approximately 30 minutes of cpu time on a Sparc10 workstation. Step 9

TABLE 4.2: STATE-FEEDBACK DESIGN: WEIGHTS/RESULTING BANDWIDTHS

iter.	W_1			W_2			W_3					ζ
	h_e	α_e	DLC	$\delta_{stab_{cmd}}$	$\delta_{thrust_{cmd}}$	$\delta_{DLC_{cmd}}$	$\dot{u}$	$\dot{w}$	$\dot{q}$	$\dot{h}$	w	
spec	$> 1s^{-1}$	$> 1s^{-1}$	$> 1s^{-1}$	$< 20s^{-1}$	$< 2s^{-1}$	$< 40s^{-1}$	-	-	-	-	-	> 0.6
1	1/ 0.4	1/ .01	1/ 0.07	1/ 150	1/ 2	1/ 2	0	0	0	0	0	0.23
2	1/ 0.4	1/ .01	1/ 0.07	5/ 150	1/ 2	1/ 2	0	0	0	0	0	0.23
3	1/ 0.4	1/ .01	1/ 0.07	50/ 50	1/ 2	1/ 2	0	0	0	0	0	0.27
4	1/ 0.4	1/ .01	1/ 0.07	100/ 40	1/ 2	1/ 2.5	0	0	0	0	0	0.27
5	1/ 0.4	1/ .01	1/ 0.07	500/ 10	1/ 2	1/ 3	0	0	0	0	0	0.27
6	1/ 0.4	1/ .01	1/ 0.07	500/ 8	0.1/ 2	1/ 3.5	0	0	0	0	0	0.27
7	1/ 0.4	100/ .01	1/ 0.07	500/ 10	0.1/ 2	1/ 3	0	0	0	0	0	0.27
8	1/ 0.4	10^4/ 2	1/ 2	500/ 20	0.1/ 2	1/ 20	0	0	0	0	0	0.27
9	1/ 0.4	10^4/ 0.5	1/ 0.4	500/ 50	0.1/ 2	1/ 2	0	0	0	0	0	0.2
10	1/ 0.4	10^4/ 0.5	1/ 0.4	10^3/ 20	0.1/ 4	1/ 2	0	0	0	0	0	0.2

marked the point when the additional integrators were added to the weighting matrix W_1. This altered several bandwidths slightly, though the scalar weights themselves were not adjusted between steps eight and nine.

The above design resulted in two pairs of lightly damped eigenvalues, with damping ratios between 0.2 and 0.4. Modal analysis revealed that only the thrust was significantly participating in these modes, with no participation by the aerodynamic states. This was regarded as acceptable, and no further damping improvement was attempted. The weighting matrix W_3 was consequently all zeros. Since the measurement-feedback controller was not pursued, the weighting matrix W_4 was left as all zeros as well.

4. Specification Compliance

Figures 4.13 and 4.14 depict the broken-loop controller responses, and the closed-loop command responses. Each of the broken-loop controller responses is less than the specification, indicating that the actuators would not be driven at frequencies greater than their bandwidths. The closed-loop command responses of 0.4 for the altitude and DLC channels were slightly less than the specification value of 0.5

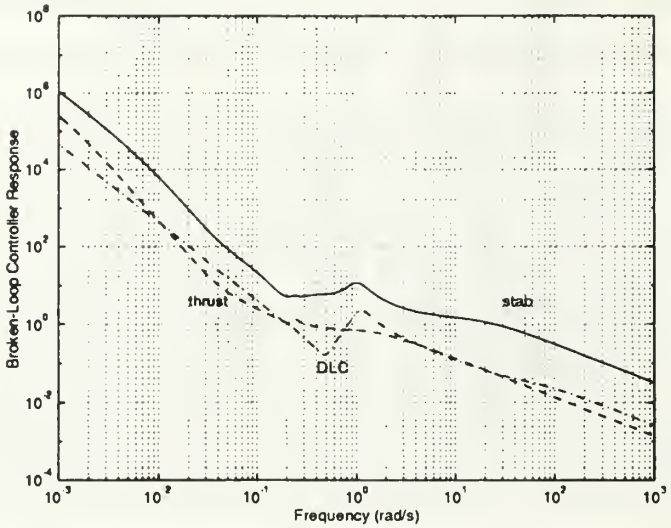

Figure 4.13: Broken-Loop Controller Responses

rad/sec. Figure 4.15 depicts the Nyquist plot for the broken-loop controller responses. The Nyquist plot shows the controller loops as satisfying the simultaneous phase and gain margins, with the exception of the DLC loop which slightly violated the outside corners. It is interesting that the DLC loop stays outside the unit circle around (-1.0), as would be expected when using Loop Transfer Recovery Methods with a pure $\mathcal{H}_2$ controller design tool.

Figure 4.16 depicts the singular values of the open-loop controller response. The clear peak corresponding to the short-period frequency indicates that this mode was not canceled by the controller.

5. μ-Analysis

Figure 4.17 depicts the structured singular values for the closed-loop system $T_{z_\delta w_\delta}(\mathcal{G}, \mathcal{C})$. The peak μ value less than one confirms the robust stability of the system to uncertainties greater than the specified 40%. That the peak value is well less than one highlights the conservative nature of using $\mathcal{H}_\infty$ constraints for robustness when the uncertainty can be expressed in a structured fashion.

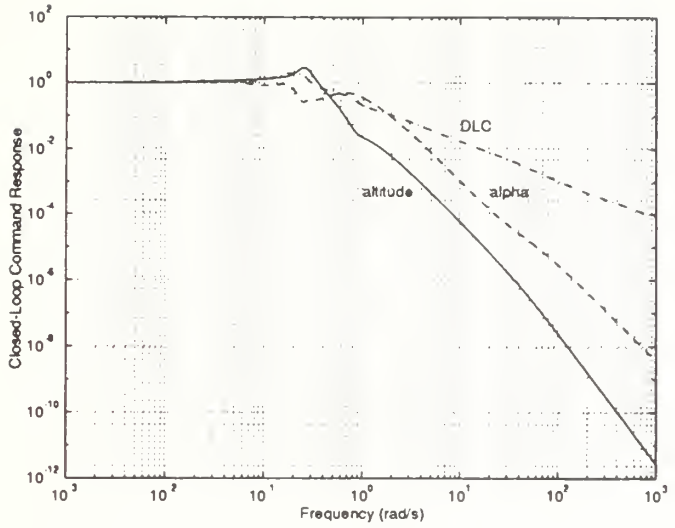

Figure 4.14: Closed-Loop Command Responses

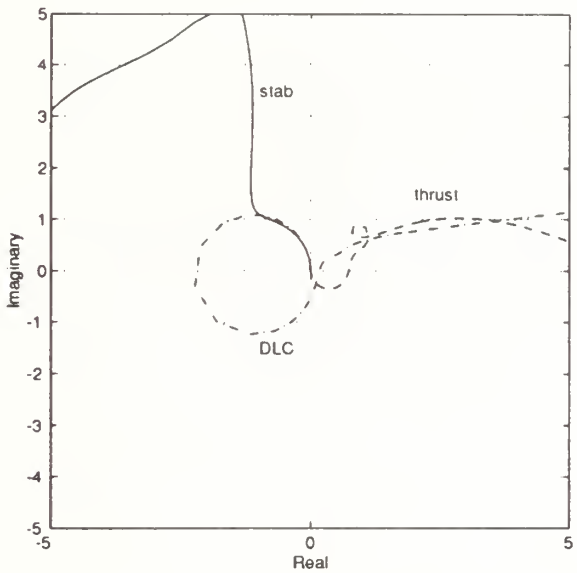

Figure 4.15: Broken-Loop Nyquist Response

6. Linear Simulation

A linear simulation was performed in order to ensure that the resulting closed-loop system fulfilled the design requirements. Figure 4.18 depicts the response of the system to a ramp altitude command. All variables are depicted as perturbations

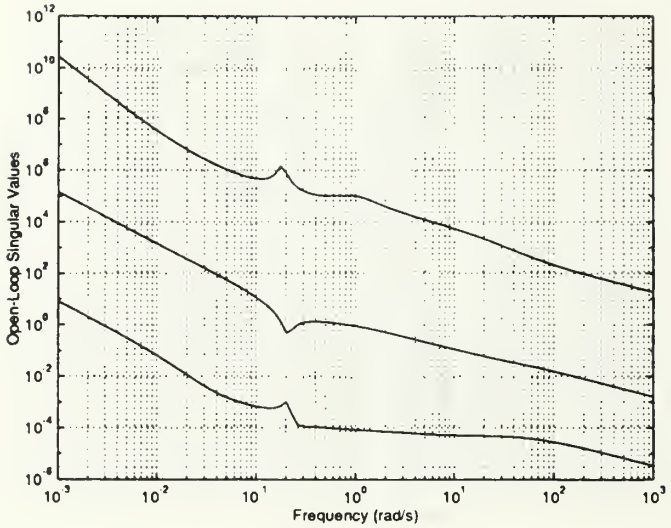

Figure 4.16: Open-Loop Singular Values

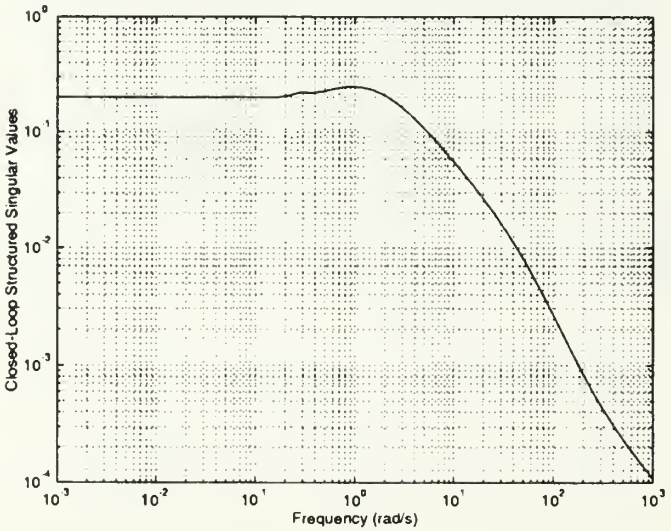

Figure 4.17: Closed-Loop Structured Singular Values

from their trimmed condition.

Since the command bandwidths are slower than those for the first design example, the response of the regulated signals to the altitude ramp is predictably slower. The desired wash-out characteristics are displayed, though, on altitude error,

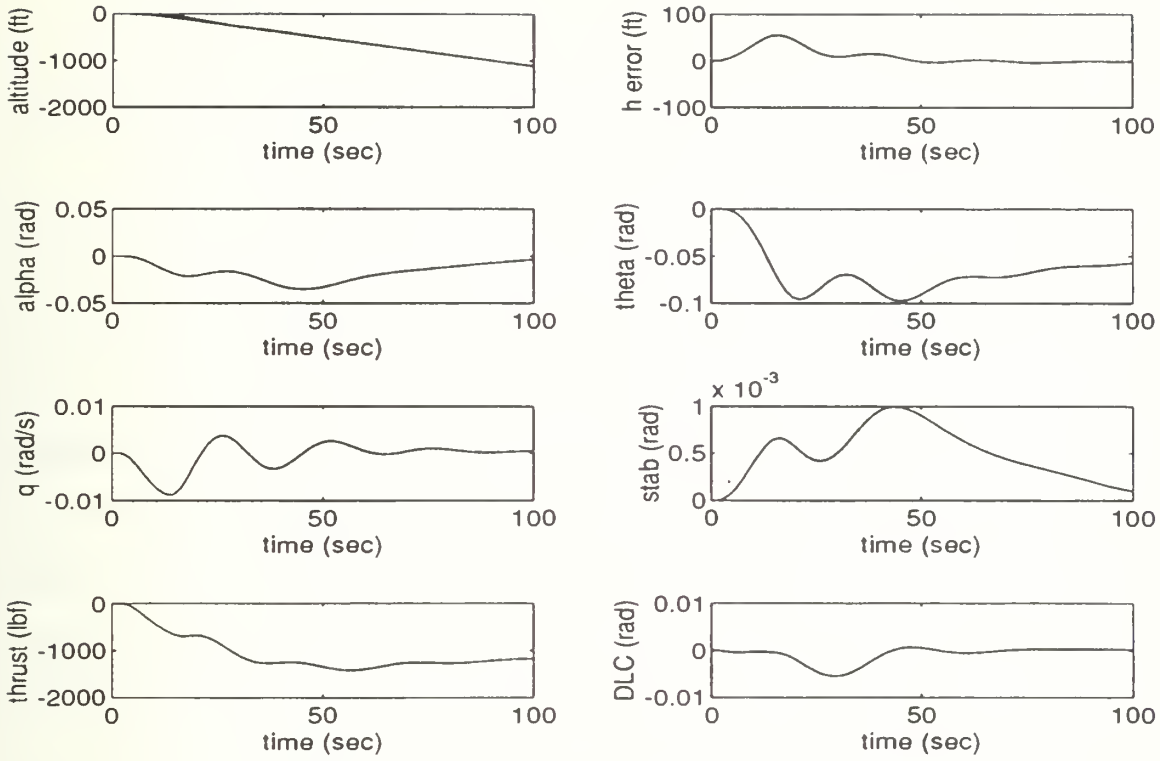

Figure 4.18: Linear Simulation Results

angle-of-attack, and DLC deflection.

7. Mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ Controller Design Conclusions

The methodology of section C. was successfully extended for use in the design of mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ state-feedback controllers. Furthermore, a synthesis model formulation was demonstrated whereby the $\mathcal{H}_2$ / $\mathcal{H}_\infty$ design tools could be applied in order to achieve explicit robustness guarantees, simultaneous with other classical SISO design requirements. The final controller design was validated by linear simulation.

E. CONTRASTING THE $\mathcal{H}_\infty$ AND $\mathcal{H}_2$ / $\mathcal{H}_\infty$ DESIGN TOOLS

Because of the similar synthesis models and design methodologies, the effort required to implement the two tools was virtually identical. In fact, a single MATLAB m-file was used to prepare the two synthesis models, with only slight variations

required. A positive attribute of the $\mathcal{H}_2$ / $\mathcal{H}_\infty$ tools was that little procedural effort was required to shift to the $\mathcal{H}_2$ / $\mathcal{H}_\infty$ tools from the $\mathcal{H}_\infty$ tools. The mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ design tools did demonstrate one significant disadvantage over the pure $\mathcal{H}_\infty$ method. While the computation of a single $\mathcal{H}_\infty$ controller would require less than 30 seconds, the computation of an $\mathcal{H}_2$ / $\mathcal{H}_\infty$ controller for this 14 state problem required between 12 and 45 minutes, depending on the scalar weights chosen. This was partially attributable to the particular implementation of the convex optimization methods. The computational time could be probably be dramatically reduced with a FORTRAN or C implementation of the interior point method, rather than the ellisoidal codes used here. However, even the most efficient optimization routine will be dramatically slower than the nearly direct computational means available by Riccati methods. Consequently, the mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ tools should only be used in situations where the $\mathcal{H}_2$ norm is an explicit expression of some specific design specification.

F. CONCLUSION

This chapter demonstrated the application of $\mathcal{H}_\infty$ and mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ controller design tools to the problem of autoland controller design. A methodology was presented whereby classical SISO design specifications could be translated into scalar weighting functions appended to the synthesis model for either controller design scheme. Furthermore, a methodology was presented for the modeling of uncertainties in flight dynamics problems, and was then applied in the analysis of the robustness of the resulting closed-loop systems.

V. INTEGRATED AIRCRAFT/CONTROLLER DESIGN BY LINEAR MATRIX INEQUALITIES

This chapter presents a methodology by which aerodynamic surface sizes can be optimized using performance requirements which are posed as Linear Matrix Inequalities. Several examples are presented which demonstrate the utility and flexibility of the method. The MATLAB files which support the material of this chapter can be found in Appendix D.

A. INTRODUCTION AND PROBLEM MOTIVATION

The control design process for rigid body vehicles (air, marine and space) includes not only the design of the control system itself, but also the refinement of the size of the control effectors. Generally, the shape and overall size of the vehicle is dictated by mission constraints such as payload, range, or maximum speed. Control effectors and stabilizing surfaces are then appended to the baseline vehicle to provide sufficient control power so that, in concert with an appropriate control design, the desired dynamic performance is realized. **Control power is very expensive**, resulting in increased weight, drag, signature, and financial cost. Consequently, the configuration designer would like to incorporate only that amount of control power that is necessary to attain the desired dynamic performance requirements. The advance of controls technology in the past 25 years have led many commercial and military aircraft designers to consider reduced static stability aircraft as a means of improving performance and lowering cost.

The question spawned by this drive toward improved performance is consequently—how much control is enough? There are actually two facets to the question. First of all, flying qualities specifications must be identified, which, if satisfied will permit the vehicle to meet its mission requirements. Much of this field relies on the subjective opinions of pools of research pilots and engineers. Flying qualities consequently translates mission requirements into dynamic performance requirements. Considerable work has been invested by the military services and NASA in recent years to expand the traditional flying qualities standards [Ref. 1] to the new paradigm of relaxed static stability aircraft. The second facet of the problem is translating the flying qualities specifications into physical configurations and control systems. This question is addressed by this work.

In aeronautical applications, control sizing is driven by several distinct environments. Each of these must be independently considered in the design, and the most stringent adopted. The three principal environments are:

- Take-off, Approach and Landing (terminal area flight);
- High angle of attack flight;
- Supersonic flight.

Corresponding to each of these environments is a set of critical center-of-gravity (*cg*) locations. For each of these *cg* locations, sufficient control power must exist to provide the controls designer with the capacity to achieve appropriate dynamic performance (stability, maneuverability, and disturbance rejection) for both nominal and failure conditions. However, the nature of the control power required to provide such performance differs greatly from environment to environment. For example, in the case of supersonic flight, the principal issue is maneuverability. Here, control power can

be degraded by changes in the aircraft center of pressure or blanking of the control surfaces by shock-waves. Experience with supersonic flight has provided a base of knowledge for providing adequate control power in this environment. Since control power is basically proportional to the airspeed squared, it is more frequently critical at the other extreme of the operating envelope, at slow speed, such as during high-angle-of-attack maneuvering and take-off/approach/landing. In the case of high angle-of-attack flight, the aerodynamics are very non-linear, and the concern is the ability to generate specific rates or accelerations (maneuverability). The specifications are therefore generally expressed as open-loop rates or accelerations.

The specifications determining the control power requirements for slow speed terminal-area flight are both closed-loop and open-loop. In this environment, the concern of the configuration designer is to provide the controls designer with adequate control power such that a controller can be designed which satisfies the specified closed-loop flying qualities, as well as open-loop maneuverability. Unlike the other two environments, linear flight mechanics prevail, and the vehicle can reasonably be modeled as a linear system. A significant portion of the control sizing problem is consequently a linear controls problem.

This chapter addresses the problem of simultaneous aircraft/controller optimization. As discussed in Chapter III, many closed-loop control problems and flying qualities specifications can be formulated as convex optimization problems, some of which can further be simplified into LMI's. It has been our observation that for many rigid body systems, performance specifications which are posed as LMI's provide a means for not only solving the controller design problem, but also directly determining the upper-bound for the minimum necessary control power. As a result of the this work, the aircraft and control system designers will be provided with a new tool capable of answering the *plant/controller optimization problem*:

Given the flying qualities requirements for a specified mission, find the minimum aerodynamic surface sizes, and a feedback controller, which together satisfy mission requirements.

The answers obtained are important to reduce aircraft weight, size, and observability and would result in cost savings in many current and future aircraft procurement programs. The application of these methods is certainly not restricted to aeronautical applications, in that control power is expensive in all engineering systems. In a chemical process system, the control power is reflected in the diameters of the pipes and tubing. The controls designer here is no less interested in minimizing the costs of achieving adequate control power.

For aeronautical applications, the seminal work to date on this subject was a design guide published by the Grumman Corporation in support of NASA's X-29 program [Ref. 5]. The scope of that effort was significant, including non-linear aerodynamics and flexible structures, issues which were beyond the scope of the material treated by this research. The authors acknowledged, however, that a limitation to their approach was the *a priori* choice of the structure of the controller. They suggested that optimization tools would be required in order to relax this constraint. The problem considered in [Ref. 5] is consequently a search over the set of plant configurations that, in concert with the specified controller, yield the desired performance requirements. Recently developed computational tools (such as those presented in Chapter II) and the convex formulation of many powerful control problems now permit the consideration of the larger problem, where the designer is not restricted to a specified controller structure, but can instead examine the set of *all* controllers which satisfy the performance measures.

The sections that follow present a formulation and methodology for the optimization of an airframe using LMI's to pose the design constraints. First, section B will define the problem. Then, in section C, a pure $\mathcal{H}_\infty$ constraint will be used to pose disturbance rejection requirments. A formulation will then be presented for the solution of the plant/controller optimization problem, and demonstrated by an example. The example problem will be a regulation problem in which there is no command tracking signal, but instead a controller which simply stabilizes the system and provides a specified disturbance attenuation. This section will also present how this formulation can be expanded to include diverse $\mathcal{H}_\infty$ specifications at multiple flight conditions. Next, in section D, both internal stability and disturbance rejection specifications will be considered. Section E will present several methodologies for formulating the various open-loop maneuverability requirements as LMI's. The examples in this section will demonstrate the application of open-loop maneuverability requirements. Next, section F presents a formulation for accommodating parametric uncertainty in plant optimization process. In wrapping up, section G discusses the results, their limitations, and other applications, while section H discusses future directions for this research.

B. PROBLEM DESCRIPTION

Once again, the problem we sought to address in this research can be stated as follows:

> Given the flying qualities requirements for a specified mission, find the minimum aerodynamic surface sizes, and a feedback controller, which together satisfy mission requirements.

It will be shown that determining an upper-bound to the optimum answer to this problem can be reduced to minimizing a linear cost (a function of aircraft parameters)

subject to matrix inequalities which represent the mission performance requirements. The problem formulation and proposed solutions are discussed next.

Let ζ be a vector of the aircraft parameters which we would like to optimize. Furthermore, suppose the plant matrices A, B_1 and B_2 can be expressed as affine functions of the plant parameters. In other words, let

$$\begin{aligned} A &= A(\zeta) = A_0 + \sum_{i=1}^{r} \zeta_i A_i, \\ B_1 &= B(\zeta) = B_{1_0} + \sum_{i=1}^{r} \zeta_i B_{1_i}, \\ B_2 &= B(\zeta) = B_{2_0} + \sum_{i=1}^{r} \zeta_i B_{2_i}. \end{aligned}$$

Later examples will demonstrate that many plant parameters, such as the area of an aerodynamic control surface, naturally occur as affine variables in a dynamic systems. Since ζ represents physical sizes for the proposed problem, we constrain $\zeta_i > 0$. Let $J = c^T\zeta$ be the cost function, where $c_i > 0$ denotes the relative cost we choose to assign to each parameter ζ_i. For example if we seek to minimize the total mass of the physical control surfaces, and ζ_i were the physical area of each control surface, then c_i might be the mass per unit area, and J the total weight of the subject components. Or c_i could be selected to be the cost per unit area, in which case J is the total cost of adding surfaces under consideration. The linear cost functional $J(\zeta)$ is clearly convex in ζ. While a linear combination of the optimization parameters is the easiest to accommodate, the general formulation proposed would permit any convex function of the optimization parameters. The general problem can now be stated as:

$$\begin{aligned} &\text{Minimize: } J = c^T\zeta, \\ &\text{Subject to: } F(\zeta, \xi) < 0, \end{aligned} \tag{5.1}$$

where $F(\zeta, \xi) < 0$ reflects all the relevant performance requirements. It was our desire to determine if these performance constraints could be posed in some manner that

would permit solution of the optimization problem by convex methods. In order to do this, we considered the various types of constraints which typically arise in flight mechanics.

The three general sets of performance criteria which drive control power requirements are stabilization, disturbance rejection, and maneuverability [Ref. 5]. Stabilization requirements pertain to the internal dynamics of the problem and are frequently posed as pole-placement requirements. Much of the concern relevant to control power optimization can be attributed to the trend towards vehicles with unstable open-loop modes, and the acknowledgment that some portion of the control power would have to be devoted to providing closed-loop stability. Simple stability can be posed as either a Lyapunov inequality, or an $\mathcal{H}_\infty$ constraint. Chapter III described how a pole-placement requirement could be posed as convex matrix inequalities.

The second type of requirement which influences the necessary control power is disturbance rejection requirements. These are usually posed as limitations on the *rms* output, as a consequence of a specified *rms* input. The disturbance attenuation specifications can be directly posed as $\mathcal{H}_\infty$ constraints, which we know to be expressible as a matrix inequality. The inputs are generally meteorological gusts, represented by a model such as the Dryden turbulence model [Ref. 1]. The outputs include both the physical response of the states and the closed-loop response of the actuators. The specification determines the first, while physical constraints, such as the actuator deflection limits and limit rates, define the limit acceptable values for the latter. The $\mathcal{H}_\infty$ constraint is well suited to these types of performance measures due to its power gain interpretation, as described in Chapter III. An $\mathcal{H}_\infty$ constraint will not ensure that a peak actuator deflection can not exceed a given value, but it can limit the power signal of the actuators deflection and rate.

Maneuverability is the third performance measure which drives control power. The constraints here have the form that a specified output should exceed a given value when all relevant controllers are fully deflected in the appropriate direction. This is an "open-loop" requirement to a flying qualities engineer because the loop in which he is interested is open— the pilot's command is saturated and the pilot is not trying to actively track some variable. This need not imply that the loops within the controller are "open", and a controls engineer may use either an open or closed-loop formulation.

Maneuverability requirements can actually have several forms. In the sizing of a rudder, for example, it may be a requirement that the minimum moment must exceed that required to balance an engine out condition. In this case, static control power is the issue. As will be shown later, this can directly be expressed as an LMI. In other applications, the specification may require that the system exhibit a minimum body rate in response to a limit command input. Since most of our controllers produce either moments or accelerations, then a specified steady state rate represents the output of a dynamic system. These types of specifications can consequently be referred to as dynamic maneuverability requirements or dynamic open-loop requirements. These too we will show to be expressible as LMI's.

In sum, control power requirements are determined by the joint requirements of closed-loop internal stability, closed-loop disturbance rejection, and open-loop maneuverability. Each of these performance constraints can be posed as matrix inequalities. The joint imposition of many such matrix inequalities provides a mechanism for solving the subject optimization problem.

C. APPLYING DISTURBANCE REJECTION REQUIREMENTS

In this section we will formulate the plant/controller optimization problem, using an $\mathcal{H}_\infty$ specification as the design constraint to apply disturbance rejection requirements, as well as simple closed-loop stability. Recall equation 3.13, which poses the $\mathcal{H}_\infty$ performance constraint as a LMI:

$$R_1(W,Y) := \begin{bmatrix} AY + YA' + B_2W + W'B_2' + B_1B_1' & (CY + DW)' \\ (CY + DW) & -\gamma^2 I \end{bmatrix} < 0, \qquad (5.2)$$

Assume that the inputs and outputs are scaled such that $\gamma = 1$, and consider ξ to be the vector of the controller parameters such that W and Y are affine functions of ξ and basis matrices Y_i, W_i: $Y = Y(\xi) = Y_0 + \sum_{i=1}^{s} \xi_i Y_i$ and $W = W(\xi) = W_0 + \sum_{i=1}^{s} \xi_i W_i$, $\xi \in \mathcal{R}^s$ and $s = (n(n+1)/2) + nq$. First, using Schur complements, equation 3.13 is equivalent to the following LMI:

$$R_2(\zeta,\xi) := \begin{bmatrix} AY + YA^T + B_2W + W^TB_2^T + (CY + DW)^T(CY + DW) & B_1 \\ B_1^T & -I \end{bmatrix} < 0. \qquad (5.3)$$

Note that this constraint is affine in the plant matrices A, B_1 and B_2.

Optimization problem 5.1 can now be exressed as:

$$\begin{aligned} &\text{Minimize } J = c^T\zeta \\ &\text{Subject to: } Y > 0,\ \zeta > 0,\ R_i(\zeta,\xi) < 0, \end{aligned} \qquad (5.4)$$

where $R_i(\zeta,\xi)$ is given by either R_1 or R_2 (equations 5.3 and 5.2). Recall, both constraints are equivalent. To simplify notation, the dependence of the matrix functions on ζ and ξ will be implicit in the sequel. The controller matrices (Y,W) will consistently be functions of ξ and the plant matrices (A,B_1,B_2) will always be functions of ζ.

As can be seen in equation 5.3, this problem is affine in ζ for a fixed controller $K = WY^{-1}$, since the plant itself is affine in ζ. On the other hand, equation 5.2

shows that for a fixed plant (A, B_1, B_2, C, D) this problem is affine in ξ . Our problem, however, is to minimize J over all feasible ζ and ξ. Even though J is affine in ζ, it is not clear whether the constraint set $\{\Phi = (\zeta, \xi) : \zeta > 0, R(\zeta, \xi) < 0\}$ is convex. This means that the numerical solution of the optimization problem 5.4 may turn out to be a local minimum. An important area of future research will be to determine whether the set Φ is convex. Presently it can be shown that for a special case of the optimization problem 5.4, the set Φ is convex. This case is discussed in the next section.

1. Special Case: Plant/Controller Optimization as a Generalized Eigenvalue Problem

Consider a single parameter plant optimization problem, where

$$\begin{aligned} A &= A_0 + \zeta A_1, \\ B_1 &= B_{1_0}, \\ B_2 &= B_{2_0} + \zeta B_{2_1}. \end{aligned} \tag{5.5}$$

Suppose the matrix pair (A_1, B_{2_1}) is stabilizable. A physical example of such an one parameter problem could be optimization of the size of the vertical tail of an aircraft, or the size of the control fin on a missile or a submersible. Now the optimization problem can be formulated as follows:

Minimize : ζ

Subject to: $Y > 0$, $\zeta > 0$, and

$$R_1 = \begin{bmatrix} \begin{pmatrix} A_0Y + YA_0' + B_2W + W'B_2' + B_1B_1' + \\ \zeta(A_1Y + YA_1' + B_{2_1}W + W'B_{2_1}') \end{pmatrix} & (CY + DW)' \\ (CY + DW) & -I \end{bmatrix} < 0. \tag{5.6}$$

Let ϵ denote a very small number. Using simple algebra and the definition of negative definiteness, it can be shown that optimization problem 5.6 is equivalent to the

following problem:

$$\text{Minimize} : \zeta$$

$$\text{Subject to: } Y > 0,\ \zeta > 0, \text{ and}$$

$$\zeta \begin{bmatrix} -(A_1 Y + Y A_1' + B_{2_1} W + W' B_{2_1}') & 0 \\ 0 & \epsilon I \end{bmatrix}$$

$$- \begin{bmatrix} A_0 Y + Y A_0' + B_2 W + W' B_2' + B_1 B_1' & (CY + DW)' \\ (CY + DW) & -I \end{bmatrix} > 0 \qquad (5.7)$$

Problem 5.7 is in the form of the Generalized Eigenvalue Problem (GEVP) (see 2.1 and [Ref. 15]):

$$\text{Minimize: } \lambda$$

$$\text{Subject to: } \lambda B(x) - A(x) > 0, A(x) = A(x)', B(x) > 0, C(x) > 0,$$

where $A(x), B(x)$, and $C(x)$ are affine in x. As mentionned in Chapter II, the GEVP is a quasi-convex problem, meaning that it has a unique global minimum which can be found using efficient numerical techniques [Ref. 14, 15].

2. General Case

Unfortunately, many of the more interesting aircraft optimization problems cannot be reduced to GEVP form. This is particularly true for the case where several aircraft parameters have to be optimized. However, the optimization problem 5.4 is affine in either the vector of plant parameters ζ, or the set of controller parameters W and Y. This suggests the following approach to solving the optimization problem 5.4: find a controller (holding the plant constant) , and then minimize the objective function while holding the controller constant:

1. Fix $A(\zeta), B(\zeta)$. Then,

$$\text{Minimize: } \lambda \text{ (over } \xi),$$

$$\text{Subject to: } \begin{bmatrix} Y(\xi) & 0 \\ 0 & \lambda I - R_1(\zeta, \xi) \end{bmatrix} > 0. \qquad (5.8)$$

2. Fix $W(\xi), Y(\xi)$. Then,

$$\begin{aligned} &\text{Minimize: } J \text{ (over } \zeta), \\ &\text{Subject to: } \begin{bmatrix} J - c^T\zeta & 0 & 0 \\ 0 & \text{diag}(\zeta) & 0 \\ 0 & 0 & -R_2(\zeta,\xi) \end{bmatrix} > 0. \end{aligned} \tag{5.9}$$

3. Go to step 1 until exit criteria is satisfied.

Denote the value of λ obtained in the first step as the *controller margin.* Given the controller determined in the first step, the second step of the procedure then finds the feasible plant with the smallest associated cost. The procedure quits when the controller margin becomes so small that the numerical procedure either fails to find a new feasible controller after step two, or step two is numerically unable to further refine the plant.

The controller margin has a very rich, significant geometrical meaning. As the maximum eigenvalue of $R_1(\zeta,\xi)$, the controller margin presents the "distance" of $R_1(\zeta,\xi)$ from singularity, and consequently the "distance" or margin by which the closed-loop system satisfies the performance specifications. It could notionally be considered to represent the "size" of the set of feasible controllers. The set of feasible controllers is either empty or infinite, so "size" is used loosely. The greater the margin, however, the greater the range of feasible controllers. This is important because the further that $R_1(\zeta,\xi)$ is from singularity, then the more "room" available for the second phase of the algorithm to optimize the plant. One would intuitively expect that as the controller margin decreases, then the amount of latitude available to optimize the plant also decreases.

Note that both steps of this algorithm are EVP's and consequently perfectly suited for solution by interior point methods. Since the problem is affine in both steps of the algorithm, the numerical procedure is guaranteed to converge. But, as

discussed earlier, whether the problem is convex jointly in the controller and plant parameters is presently not known and is subject of future research. Therefore, there is no assurance that the obtained minimum is global (unless the problem satisfies the conditions of the special case). The above procedure clearly determines a valid upper bound for the joint procedure, since a solution by interior point methods would never leave the joint feasible set. We have not, however, established a means for computing a lower-bound. This too is an issue to be pursued by further research.

A sequence of examples will demonstrate the viability of the proposed methodology and illustrate several of its more powerful implications. For the first several examples, a single plant variable is optimized. The examples build in complexity as the methodology is expanded to accommodate the other various performance measures.

Example One- Optimal Vertical Tail at a Single Flight Condition

This very simple example will demonstrate the formulation of the problem, as well as some of the properties that were discussed above. Consider the directional dynamics of a typical fighter-size aircraft. The most simple approximation is a second order system, with aerodynamics providing both restorative and dissipative forces. The wing-body combination of most aircraft is sized and shaped by factors such as range, maximum speed, payload, powerplant. Typically, the directional dynamics of the wing-body combination include both a stable and an unstable pole. Given a specific wing-body combination, the problem for the aero configuration designer and control designer is to determine the size of the vertical tail and controller, which, together in feedback, provide satisfactory dynamic behavior. For simplicity, we'll consider an all-moving tail so as to reduce the number of degrees of design freedom

to one. Because of the single degree of freedom, this problem is quasi-convex and is know *a priori* to have a unique, global minima.

The governing differential equation is:

$$\ddot{\beta} = \frac{qSb}{I_z}\left[-C_{n_\beta}\beta - \frac{2l_t}{V}C_{n_\beta}\dot{\beta} + C_{n_\beta}\frac{w}{V} - C_{n_\delta}\delta\right]. \tag{5.10}$$

The notation here is standard aeronautical usage:

$$\begin{array}{rcl} w & := & \text{disturbance input} \\ \delta & := & \text{control deflection} \\ \beta & := & \text{sideslip angle} \\ q & := & \text{dynamic pressure} \\ S & := & \text{wing reference area} \\ b & := & \text{wing span} \\ I_z & := & \text{moment of inertia about the vertical } (z) \text{ axis} \\ l_t & := & \text{vertical tail lever arm (distance from CG to aero center of tail)} \\ C_{n_\beta} & := & \text{non-dimensional change in yawing moment per radian of sideslip} \\ V & := & \text{aircraft velocity} \end{array} \tag{5.11}$$

The contribution of the tail and wing/body can be separated:

$$C_{n_\beta} = C_{n_{\beta_{wb}}} + C_{l_{\beta_t}}\mathcal{V} \tag{5.12}$$

where $\mathcal{V} = \frac{A_t l_t}{Sb}$ is defined as the vertical tail volume. Note that $\mathcal{V}$ is a dimensionless, affine function of the vertical tail area A_t (or l_t!). This is the quantity which we shall optimize. Using 5.12, 5.10 can be rewritten as follows:

$$\begin{bmatrix} \dot{\beta} \\ \ddot{\beta} \end{bmatrix} = A\begin{bmatrix} \beta \\ \dot{\beta} \end{bmatrix} + B_1 w + B_2 \delta, \tag{5.13}$$

where:

$$\begin{array}{rcl} A & = & A_{wb} + A_t\mathcal{V} \\ & = & \begin{bmatrix} 0 & 1 \\ -\frac{qSbC_{n_{\beta_{wb}}}}{I_z} & -\frac{2Sbl_tC_{n_{\beta_{wb}}}}{I_zV} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ -\frac{qSbC_{l_{\beta_t}}}{I_z} & -\frac{2qSbl_tC_{l_{\beta_t}}}{I_zV} \end{bmatrix}\mathcal{V}, \\ B_1 & = & B_{1_{wb}} + B_{1_t}\mathcal{V} \end{array} \tag{5.14}$$

$$= \begin{bmatrix} 0 \\ -\frac{qSbC_{n_{\beta_{wb}}}}{I_z} \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{qSbC_{l_{\beta_t}}}{I_z} \end{bmatrix} \mathcal{V} \tag{5.15}$$

$$B_2 = B_{2_{wb}} + B_{2_t}\mathcal{V}$$

$$= \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{qSbl_tC_{l_{\beta_t}}}{I_z} \end{bmatrix} \mathcal{V} \tag{5.16}$$

Note that for this problem, A_1 has eigenvalues at zero and $-\frac{2qSbl_tC_{l_{\beta_t}}}{I_z V}$ $(C_{l_{\beta_t}} > 0)$.

The following numerical values were obtained using stability derivative data from [Ref. 37] and [Ref. 40] for a fighter aircraft at a flight condition of 230 fps at sealevel.

$$A = \begin{bmatrix} 0 & 1 \\ 0.0103 & 0.0018 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ -4.1044 & -0.7067 \end{bmatrix} \mathcal{V}$$

$$B_1 = \begin{bmatrix} 0 \\ 0.0004 \end{bmatrix} + \begin{bmatrix} 0 \\ -0.1765 \end{bmatrix} \mathcal{V}$$

$$B_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ -4.1044 \end{bmatrix} \mathcal{V}.$$

The tail volume $\mathcal{V}$ was initialized at a value of 0.47.

The problem was to determine the minimum tail volume which together with the feedback controller would stabilize the plant, and would limit the rms actuator deflection and sideslip to ten and two degrees, respectively, in the presence of moderate turbulence. MIL-8785C establishes moderate turbulence as 10 fps at sealevel. B_1 is scaled by a factor of $10fps/230fps$ to normalize the rms value of w to 1 radian. Since the outputs of interest are β and δ, the synthesis model is completed by:

$$y = \begin{bmatrix} \beta \\ \delta \end{bmatrix} = \begin{bmatrix} 57.3/10 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \beta \\ \dot{\beta} \end{bmatrix} + \begin{bmatrix} 0 \\ 57.3/2 \end{bmatrix} \delta. \tag{5.17}$$

The output signals have likewise been scaled to guarantee that an rms output of 1 rad satisfies the performance requirements.

An interior point method was used to perform the optimization problem of posed by algorithm 5.8 and 5.9 to determine: (a) the smallest vertical tail vol-

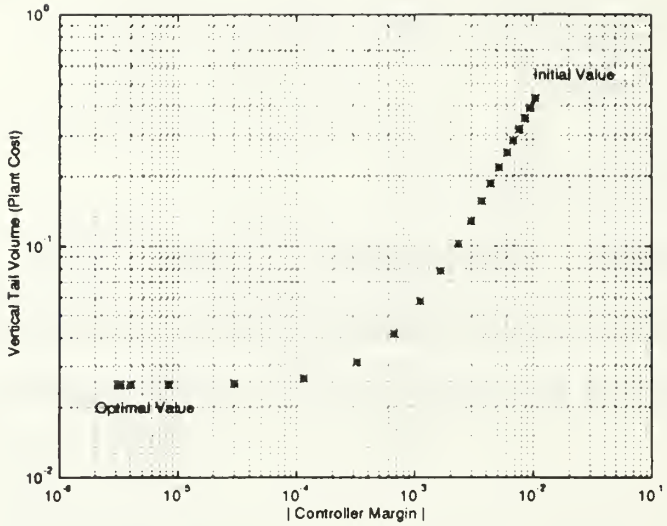

Figure 5.1: Optimization History for Tail Volume (Example One)

ume, and (b) the associated state-feedback controller which satisfied the performance requirements. Recall that the design of a controller which satisfies an $\mathcal{H}_\infty$ bound implicitly satisfies internal stability. The final required tail volume was $\mathcal{V} = 0.0250$, and the associated controller was $K = [5.443 \;\; 10.467]$.

Figure 5.1 depicts the progress of the optimization algorithm, with the algorithm commencing in the top right corner. Note that the cost and the controller margin decreased monotonically with each iteration. For a single variable problem, the weight was set to one, so that the plant cost was identically equal to the tail volume. The algorithm forces the cost to decrease monotonically since the terminal cost at the end of each step is then the starting cost for the commencement of the next iteration. For a single plant variable, the controller margin would be expected to decrease monotonically, since the margin reflects the size of the set of feasible controllers. Since A_{wb} was not stable, then the open-loop poles predictably moved to the right as $\mathcal{V}$ was decreased. Consequently, as the poles of the open-loop plant shifted to the right, the set of feasible controllers would be expected to monotonically shrink

as well (this was not true for multi-variable problems, as will be shown later). The process was terminated when the numerical routines either 1) could not find a feasible controller during step 1; or 2) the controller margin exiting step 2 was too small to numerically permit further refinement of the plant. In this particular example, the graph clearly depicts the minimum bound achievable by this method.

This example clearly demonstrated the viability of the general method. It was, however, limited in its applicability, since it only provided the size of the tail volume required at a single point in the operating environment, and the fulfillment of a single performance criteria. The challenge for the configuration designer is the determination of the optimal plant which satisfies diverse performance requirements at all points within the operating envelope. The next section will show how the method easily expands to accommodate additional constraints.

3. Plant/Controller Optimization with Multiple Joint $\mathcal{H}_\infty$ Constraints

Consider now the problem of finding the optimum size of the vertical tail which will satisfy three different $\mathcal{H}_\infty$ performance requirements at distinct flight conditions. Many of the plant variables in equation 5.16 are dependent on the flight condition, including the moment of inertia, the dynamic pressure and the stability derivatives. The plant matrices may consequently be dramatically different. Let $\mathcal{G}_1(\zeta)$, $\mathcal{G}_2(\zeta)$, and $\mathcal{G}_1(\zeta)$ represent the aircraft dynamics of a single configuration, but at three separate flight conditions. Let ξ, $\tilde{\xi}$ and $\hat{\xi}$ represent the controllers that satisfy the three distinct $\mathcal{H}_\infty$ performance requirements represented by $R_1(\zeta,\xi) < 0$; $R_2(\zeta,\tilde{\xi}) < 0$; and $R_3(\zeta,\hat{\xi}) < 0$. These three independent $\mathcal{H}_\infty$ requirements will be satisfied if :

$$R(\zeta,\xi,\tilde{\xi},\hat{\xi}) = \begin{bmatrix} R_1(\zeta,\xi) & 0 & 0 \\ 0 & R_2(\zeta,\tilde{\xi}) & 0 \\ 0 & 0 & R_3(\zeta,\hat{\xi}) \end{bmatrix} < 0. \tag{5.18}$$

The optimization problem can therefore be formulated as follows:

$$\text{Minimize: } J = c^T \zeta,$$

$$\text{Subject to: } R(\zeta, \xi, \tilde{\xi}, \hat{\xi}) < 0,\ \zeta > 0.$$

This problem is affine in the plant variable ζ for fixed controllers, and affine in the controller variables $\xi, \tilde{\xi}$, and $\hat{\xi}$ for a fixed plant. The solution to this problem is the optimal plant which permits satisfaction of each of the three performance requirements, and each of the three associated controllers. The problem strategy is similar to that pursued above with only minor modification. Since the three controller solutions are independent, they can be determined independently to save computational cost, with the plants fixed.

Example Two- Optimal Vertical Tail at Multiple Flight Conditions

To demonstrate the multiple flight condition problem, the following three flight conditions were chosen for the aircraft in the Example 1: 230 fps at sealevel, 876 fps at 35 Kft, and 1742 fps at 55 Kft. These conditions correspond to the approach, subsonic cruise, and supersonic dash mission phases. Models were again synthesized from [Ref. 37] and [Ref. 40]. The horizontal gust magnitude was 10 fps at sealevel, and 5 fps at the other two conditions. Identical constraints on the performance outputs were imposed (2 degrees of rms sideslip and 10 degrees of rms tail deflection).

Figure 5.2 depicts the progress of the algorithm. Note that one flight condition dominates the process, and the algorithm quits when the controller margin is exhausted at the slow speed flight condition. Intuition is confirmed in that for a single variable problem, one would expect the limiting tail volume to be identical to the greatest of the three tail volumes when calculated independently. The three

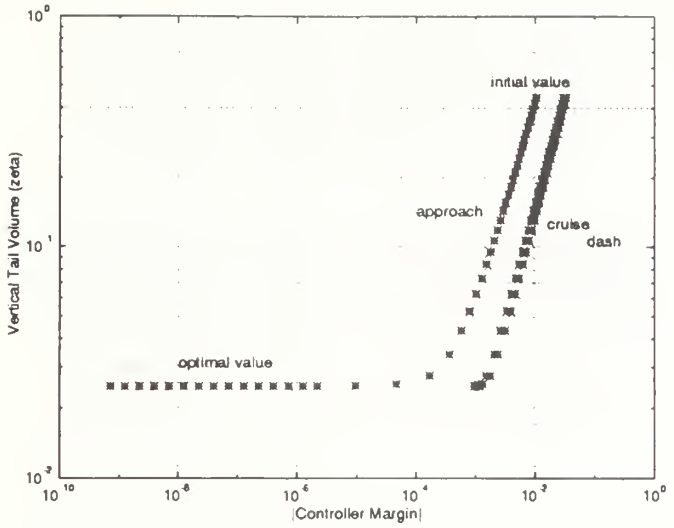

Figure 5.2: Optimization History Multiple Flight Conditions (Example Two)

resulting controllers were: $K_{slow} = [5.443 \quad 10.467]$. $K_{subsonic} = [31.396 \quad 33.350]$, and $K_{supersonic} = [38.647 \quad 37.949]$. This single procedure therefore results in an optimal plant configuration and the necessary controller for each operating point.

The physical structure of our methodology thereby accommodates the simple inclusion of multiple plant operating conditions by direct diagonal augmentation of the constraint matrix functional. The number of flight conditions is limited only by one's patience in waiting for the computational outcome, and the ability of the numerical routines to find viable answers.

D. APPLYING JOINT DISTURBANCE REJECTION/ STABILIZATION REQUIREMENTS

Now consider a problem in which the disturbance rejection requirement is imposed, along with a requirement that the closed-loop poles be placed in a specified circle. The $\mathcal{H}_\infty$ norm will again be used to impose the disturbance rejection requirement. Note that minimizing the control power subject to a pole-placement require-

ment is not by itself an interesting problem. Intuitively, some constraint on actuator activity is mandatory in the problem formulation. Otherwise, the solution would simply allow actuator activity (represented by the state-feedback gain) to increase in an unbounded fashion to compensate for the decrease in control power.

Let α and r define a circle of radius r centered at $q = -(\alpha + r)$. As discussed in Chapter III, the poles of $(A + B_2K)$ are in the circle defined by α and r, if and only if there exists a positive definite solution $Y = Y^T > 0$ such that the following matrix inequality is satisfied:

$$S(\zeta, \xi) \quad < \quad 0 \tag{5.19}$$

where,

$$\begin{aligned} S(\zeta, \xi) \quad := \quad & (A + B_2K + \alpha I)Y + Y(A + B_2K + \alpha I)^T \\ & +(A + B_2K + \alpha I)(Y/r)(A + B_2K + \alpha I)^T. \end{aligned} \tag{5.20}$$

Using the substitution $K = WY^{-1}$, this expression is equivalent to both of the following matrix inequalities, by Schur complements:

$$S_1(\zeta, \xi) := \begin{bmatrix} \left(\begin{array}{c} (A + \alpha I)Y + Y(A + \alpha I)^T + (A + \alpha I)(Y/r)(A + \alpha I)^T + \\ B_2W((A + \alpha I)/r + I)^T + ((A + \alpha I)/r + I)W^TB_2^T \end{array} \right) & B_2W \\ (B_2W)^T & -Yr \end{bmatrix} < 0 \tag{5.21}$$

and

$$S_2(\zeta, \xi) := \begin{bmatrix} (A + B_2K + \alpha I)Y + Y(A + B_2K + \alpha I)^T & (A + B_2K + \alpha I) \\ (A + B_2K + \alpha I)^T & -\mathrm{inv}(Y/r) \end{bmatrix} < 0 \tag{5.22}$$

The $S_1(\zeta, \xi)$ is affine in ξ, and $S_2(\zeta, \xi)$ is affine in ζ, but neither is affine in both. Since the two expressions are equivalent we can alternatively use them for the controller design and the plant optimization. The mechanics of constraining the problem to

simultaneously satisfying both the pole placement requirement and the $\mathcal{H}_\infty$ bound, is analogous to the multiple flight condition problem described previously. The problem can be stated as:

$$\begin{aligned} &\text{Minimize: } J = c^T\zeta \\ &\text{Subject to: } T(\zeta,\xi) := \begin{bmatrix} R(\zeta,\xi) & 0 \\ 0 & S(\zeta,\xi) \end{bmatrix} < 0, \text{and } \zeta > 0. \end{aligned} \tag{5.23}$$

Here, we again employ our prior approach of dividing the problem into two LMI sub-problems. Define two matrix inequalities:

$$T_1(\zeta,\xi) := \begin{bmatrix} R_1(\zeta,\xi) & 0 \\ 0 & S_1(\zeta,\xi) \end{bmatrix} < 0 \tag{5.24}$$

$$T_2(\zeta,\xi) := \begin{bmatrix} R_2(\zeta,\xi) & 0 \\ 0 & S_2(\zeta,\xi) \end{bmatrix} < 0 \tag{5.25}$$

We now propose the following algorithm:

1. With ζ fixed,

$$\begin{aligned} &\text{Minimize: } \lambda, \text{ (over } \xi) \\ &\text{Subject to: } \begin{bmatrix} \lambda I - T_1 & 0 \\ 0 & Y \end{bmatrix} > 0. \end{aligned}$$

2. With optimal ξ from step 1,

$$\begin{aligned} &\text{Minimize: } J, \text{ (over } \zeta) \\ &\text{Subject to: } \begin{bmatrix} J - c^T\zeta & 0 & 0 \\ 0 & \text{diag}(\zeta) & 0 \\ 0 & 0 & -T_2 \end{bmatrix} > 0. \end{aligned}$$

3. Iterate until termination criteria satisfied.

Due to the affine problem formulation, both steps may be solved by either interior point or convex methods. The $\mathcal{H}_\infty$ constraint and the pole placement constraint are just two examples of constraints that can be expressed as LMI's. Consequently,

this method of diagonally augmenting constraints to achieve a joint constraint can be used with as many jointly feasible diverse constraints as desired. Examples might include the intersection of several circles, or distinct $\mathcal{H}_\infty$ $\mathcal{H}_\infty$ bounds. We will later formulate the $\mathcal{H}_\infty$ / $\mathcal{H}_\infty$ problem in providing for distinct $\mathcal{H}_\infty$ bounds on robustness and on performance.

Example Three– Vertical Tail Optimization with Joint Constraints ($\mathcal{H}_\infty$ and Pole Placement)

Consider again the single flight condition problem from example 1, retaining the disturbance-to-output performance criteria. MIL-8785C requires that the closed-loop dutch-roll frequency exceed 1 rad/sec and that the damping ratio exceed 0.15. These requirements can be (conservatively) satisfied by constraining the poles to a circle of radius 87.3, and centered at (-88.3,0). By jointly posing these constraints using the above formulation, the plant optimization problem was solved using interior point methods. The resulting optimal tail volume was $\mathcal{V} = 0.0383$, with an associated controller $K = [15.96 \quad 18.91]$. The resulting poles were at $-1.500 \pm 0.637j$ (inside the circle). Figure 5.3 depicts the progress of the algorithm. Note that the addition of the pole placement constraint required a larger tail volume than was achieved in Example 1 for the $\mathcal{H}_\infty$ constraint alone, where $\mathcal{V} = 0.0253$ was required.

A significant attribute of this problem is that unlike the previous two, this example (or any problem with joint constraints) could not have been be solved by bisection on ζ and a standard Riccati based $\mathcal{H}_\infty$ controller design. We have now demonstrated how the methodology accomodates both stabilization and disturbance rejection specifications. Next, we will demonstrate the inclusion of open-loop maneuverability specifications into the problem methodology.

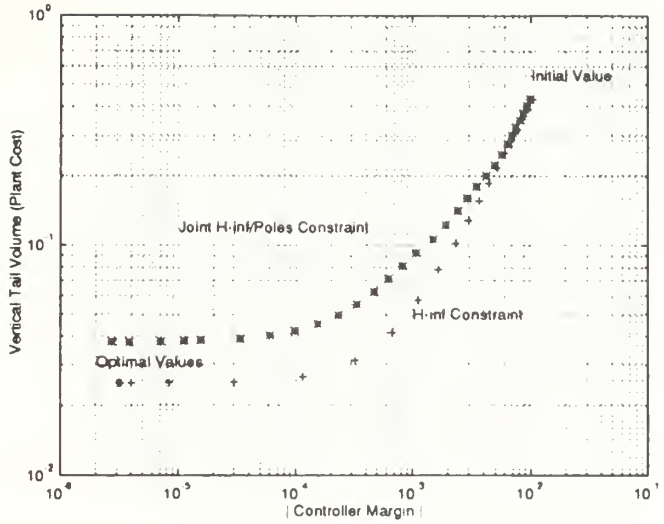

Figure 5.3: Optimization History for Joint $\mathcal{H}_\infty$ / Pole-Placement (Example Three)

E. INCLUDING MANEUVERABILITY REQUIREMENTS

This section presents several methodologies by which open-loop maneuverability requirements can be expressed as LMI's. Maneuverability requirements are very much unlike disturbance rejection, where the closed-loop plant must demonstrate a maximum acceptable level of attenuation in the presence of disturbances. Instead, maneuverability requirements demand that a system demonstrate a *minimum amplification* in the presence of a command signal. Consider the open-loop system depicted by Figure 5.4. A typical maneuverability specification has the following form: *given a maximum control input* $u = u_{max}$, *the steady state response of a scalar output* r *must exceed a certain threshold,* r_{thres} . Note that u may be a vector of all relevant control surfaces.

Alternatively, consider the closed-loop system of Figure 5.5. The maneuverability specifications above could equivalently be expressed in the context of this closed-loop system: *Given an input reference signal* r_{cmd} *of magnitude* r_{thres}, *the out-*

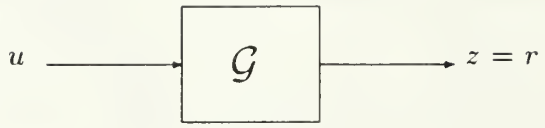

Figure 5.4: Open-loop Formulation for Maneuverability Constraints.

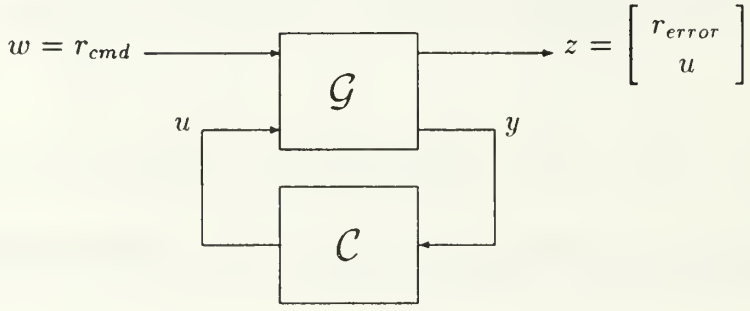

Figure 5.5: Closed-Loop Formulation for Maneuverability Constraints.

put $r_{error} = r - r_{cmd}$ *must be zero in steady state, and the control inputs* u_i *must not exceed* u_{max_i} *in steady state.*

Maneuvering specifications fall into two general classes, for which each of these formulations are helpful in posing the requirements as LMI's. As examples, consider two different longitudinal maneuverability specifications for a tactical aircraft with canards, stabilators and thrust vectoring as longitudinal control effectors. In the first case, sufficient control power might be required to provide at least 9 g's at a specific flight condition, with all effectors fully deflected or deflected no further than some effective limit (aerodynamic surfaces may lose effectiveness long before reaching a physical actuator limit). Alternatively, a requirement might be that the sum of the effectors generate a pitch acceleration of at least 0.25 rad/s at full deflection at a specified flight condition. Both of these types of design specifications exist. The output r in either case would be scalar, while the input u would be a vector of the

three control effectors.

To understand the difference between these two types specifications, consider the state space representation of the open-loop system depicted by Figure 5.4:

$$\mathcal{G} = \begin{cases} \dot{x} &= A(\zeta)x + B_2(\zeta)u \\ z &= C(\zeta)x + D(\zeta)u \\ r &= z \end{cases} . \tag{5.26}$$

Let T_{open} denote the transfer matrix from u to r in Figure 5.4. Then it has the following form:

$$T_{open} : r(s) = \left\{C(\zeta)(sI - A(\zeta))^{-1}B_2(\zeta) + D(\zeta)\right\} u(s). \tag{5.27}$$

For the examples we cited above, if r is the pitch acceleration, then D is non-zero, and C is zero, annulling the first term of the transfer function matrix. On the other hand, if r is the load factor, then the D matrix of the above open-loop system is zero. The first case can be considered a *static* maneuverability specification, while the second one a *dynamic* maneuverability specification. We will see that a static maneuverability requirement is easily posed as an LMI. In the case of dynamic specifications, however, both the closed-loop and open-loop formulations can be useful.

1. Static Maneuverability Requirements

First, let us consider the static maneuverability requirements. These include constraints where the modeled open-loop system has a non-zero D and zero C matrices. An LMI formulation for the requirement is evident by inspection:

$$D(\zeta)u_{max} - r_{thres} > 0. \tag{5.28}$$

Multiple similar requirements could be diagonally stacked. This type of specification can also referred to a static control power constraint, because it frequently is expressed as the static moment required from a controller or set of controllers at a specified condition. These conditions can either be equilibrium or non-equilibrium.

The specification itself can have units of force or moment, or translational or angular acceleration in response to a limit control deflection. The next example will demonstrate how specifications of this type can simply be accommodated by our methodology.

Example Four- Vertical Tail Optimization with Static Moment Requirements

A specification that frequently sizes vertical control surfaces is the requirement that adequate control power be available to balance the moments generated by adverse thrust conditions, such as engine out. Consider the vertical tail sizing problem discussed above. In addition to the previous $\mathcal{H}_\infty$ specification for turbulence rejection and stabilization, let us now also impose the static requirement that the tail be sized so as to be able to generate a moment adequate to balance 40,000 $ft - lb$ of torque at full deflection (30 degrees) at our nominal flight condition. This requirement is representative of the torque necessary to balance an asymmetric thrust condition. Furthermore, it can simply be posed as the following LMI:

$$D_{static}(\mathcal{V}) := qSbu_{max}\mathcal{V} - r_{thres} > 0, \tag{5.29}$$

where $u_{max} = 0.52\ rad$, and $r_{thres} = 40,000\ ft - lb$.

Let $\xi = \mathcal{V}$. Since the constraint $D_{static}(\xi) > 0$ is not relevant to the search for a feasible feedback controller, it does not need to be considered during the controller optimization phase of the process. Therefore, the algorithm can now be posed as follows:

1. With ζ fixed,

$$\begin{aligned} &\text{Minimize: } \lambda, \text{ (over } \xi) \\ &\text{Subject to: } \begin{bmatrix} \lambda I - R_1 & 0 \\ 0 & Y \end{bmatrix} > 0. \end{aligned}$$

2. With optimal ξ from step 1,

$$\text{Minimize: } J, \text{ (over } \zeta)$$

$$\text{Subject to: } \begin{bmatrix} J - c^T\zeta & 0 & 0 & 0 \\ 0 & \text{diag}(\zeta) & 0 & 0 \\ 0 & 0 & -R_2 & 0 \\ 0 & 0 & 0 & D_{static}(\zeta) \end{bmatrix} > 0.$$

3. Iterate until termination criteria satisfied.

Figure 5.6 depicts the progress of the algorithm, with the starting point at the upper-right end of the trace. This graph depicts two significant issues. First of all, the final required value was $\mathcal{V} = 0.14$, rather than the value of 0.025 which was required when the $\mathcal{H}_\infty$ constraint alone was applied. This answer is in fact identical to that obtained if we had alternatively solved the linear equation:

$$qSbu_{max}\mathcal{V} - r_{thres} = 0.$$

Secondly, the controller margin at the conclusion of the optimization is several orders of magnitude greater than that for any of the previous problems. This indicates that the set of feasible controllers is comparatively large, and that the final answer was independent of closed-loop dynamic issues and was instead driven by the open-loop maneuverability requirement. This is consistent with the industry's practical experience that directional control power is generally sized by considerations such as thrust asymmetry rather than turbulence rejection.

The kink in the plot for the joint specification is caused by the nature of the graph. The x-axis reflects the result of the controller optimization phase of the algorithm, while the y-axis reflects the result of the plant optimiztion. The static maneuverability constraint is only applied during the latter phase, and so when the static constraint boundary is reached, the controller phase finds a new controller with

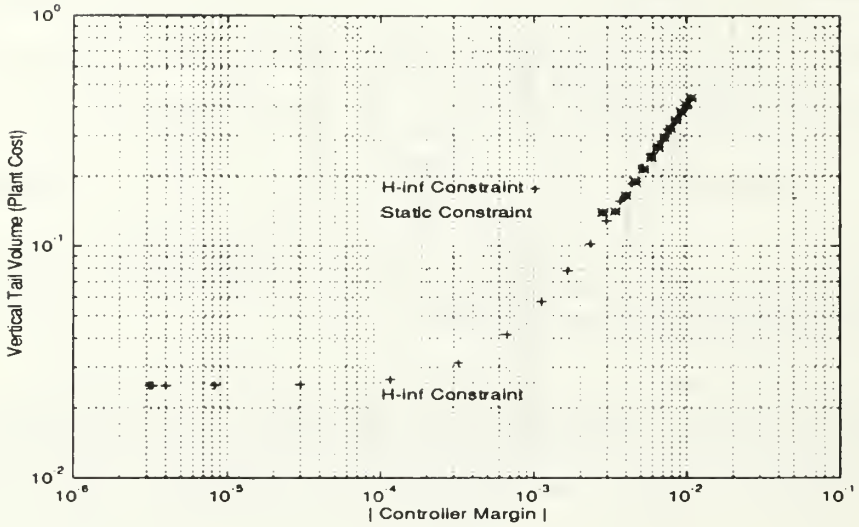

Figure 5.6: Optimization History for Vertical Tail Volume with Static Constraint (Example Four)

a smaller control margin, but the plant optimization phase can not decrease the plant cost any further. The result is an abrupt kink in the plot.

We have consequently demonstrated how static maneuverability specifications can be accommodated into the methodology.

2. Dynamic Maneuverability Requirements

Now consider the problem of posing dynamic maneuverability specifications, where for the open-loop system $D = 0$, and C is a rank one row vector. As mentionned above, these types of requirements can be posed either in an open-loop or closed-loop formulations.

a. Dynamic Maneuverability Requirements: Closed-Loop Formulation

We will first show how a maneuverability specification can be posed as an LMI by means of the closed-loop formulation. Let $T_{u,w}(\mathcal{G}, \mathcal{C})$ represent the closed loop transfer function from exogeneous disturbance $w = r$ to the control command

u_i. Here we are interested in the steady state response of the control command u_i to the constant disturbance input $w = r_{thres}$. In particular, absolute value of u_i must not exceed u_{max_i} in the presence of $w = r_{thres}$:

$$\left.\frac{u_i(s)}{w(s)}\right|_{s=0} \leq \frac{u_{max_i}}{r_{thres}}. \tag{5.30}$$

Two approaches were considered for formulating this constraint as an LMI.

(1) An $\mathcal{H}_\infty$ Approach

First, using the power interpretation of the $\mathcal{H}_\infty$ norm for a stable SISO system, this requirement can be rewritten as a constraint on the $\mathcal{H}_\infty$ norm of $T_{u_i w}(\mathcal{G}, \mathcal{C})$:

$$\left.\frac{u_i(s)}{w(s)}\right|_{s=0} \leq \|T_{u_i w}(\mathcal{G}, \mathcal{C})\|_\infty \leq \frac{u_{max_i}}{r_{thres}}. \tag{5.31}$$

Notice, if the $\mathcal{H}_\infty$ norm of $T_{u_i w}(\mathcal{G}, \mathcal{C})$ occurs at a frequency other than zero, the constraint 5.31 provides only a sufficient condition for meeting the requirement 5.30. However, it has been our experience that for the class of problems involving integral control, the $\mathcal{H}_\infty$ norm of $T_{u_i w}(\mathcal{G}, \mathcal{C})$ occurs at the origin.

Consequently, the $\mathcal{H}_\infty$ specification previously applied for disturbance rejection can now be also applied to meet dynamic maneuverability requirements. Depending on the problem, these requirements must be satisfied at the same flight condition as the disturbance rejection requirements (by including the command signal in the exogenous input vector w), or at a different more critical for maneuverability condition.

Example Five– Optimization of Multiple Surfaces with Open and Closed-Loop Performance Requirements

This example problem has several objectives. First of all, it is a problem where the cost function is dependent upon multiple plant parameters and

thus provides a much more rigorous test of the proposed methodology. Secondly, a multiple variable problem permits using a variety of the initial values for the plant parameters. This will help determine whether the algorithm will terminate at different answers, perhaps giving some insight into the convexity of the multiparameter problem. Recall, the problem is convex in the single parameter case. Finally, a problem was needed with both open and closed-loop performance requirements.

Because of the availability of component stability derivative data for the F-14 aircraft, a longitudinal control problem similar to the one in Chapter IV was selected. The vehicle parameters to be optimized were the normalized control powers of both the horizontal stabilators, and the direct lift control (DLC). Consequently, let the vector of plant parameters ζ be defined as:

$$\zeta = [\zeta_1, \ \zeta_2]^T := \left[\frac{C_{L_{stab}}}{C_{L_{stab_{nominal}}}}, \ \frac{C_{L_{DLC}}}{C_{L_{DLC_{nominal}}}} \right]^T .$$

The cost function weights on the plant parameters were arbitrarily chosen to be $c = [3, \ 1]^T$. Recall, these weights can represent normalized cost in dollars, weight in pounds, etc. The reference input of interest was commanded flight path angle γ, and the outputs to be regulated were the actuator deflections, the angle of attack error, and the flight path angle error. The disturbance input was a vertical gust. The control inputs were stabilators and DLC deflection (thrust was assumed constant), and the full state vector was assumed to be available for feedback. Thus, the problem was stated as:

> Find the plant parameters (ζ) and a state-feedback controller (ξ) which minimize the total cost $J = c^T\zeta$ of the longitudinal control effectors, subject to the following dynamic requirements:
>
> - Step response— The controller must track a step flight path angle command, γ_{cmd}, with no steady state error in α or γ.

- Closed-loop stability— The closed-loop system must be stable.
- Closed-loop performance— In the presence of a vertical gust disturbance, w_{gust}, with a magnitude of 5 fps, the stabilator deflection should not exceed 20 degrees, the DLC deflection should not exceed 40 degrees, and the angle-of-attack error should not exceed 1.5 degrees (all quantities rms).
- Open-Loop maneuverability— The plant must be able to generate a flight path angle ($\gamma = \theta - \alpha$) of 3 degrees with the DLC and stabilator deflected no more than 40 degrees and 20 degrees, respectively.

These requirements can be satisfied by the joint imposition of the following $\mathcal{H}_\infty$ constraints:

1. The step response requirement will be satisfied if and only if

$$\|T_{z_1 \gamma_{cmd}}(\mathcal{G}, \mathcal{C})\|_\infty < \infty, \text{ where } z_1 := [\frac{\alpha_{error}}{s}, \frac{\gamma_{error}}{s}]^T.$$

2. The closed-loop performance will be satisfied if and only if

$$\|T_{z_2 w_{gust}}(\mathcal{G}, \mathcal{C})\|_\infty < \frac{1}{5 fps}, \text{ where } z_2 := \left[\frac{\delta_{stab}}{20\ deg}, \frac{\delta_{DLC}}{40\ deg}, \frac{\alpha}{1.5\ deg}\right]^T.$$

3. The open-loop maneuverability requirement will be satisfied if

$$\|T_{z_3 \gamma_{cmd}}(\mathcal{G}, \mathcal{C})\|_\infty < \frac{1}{3\ deg}, \text{ where } z_3 := [\frac{\delta_{stab}}{20\ deg}, \frac{\delta_{DLC}}{40\ deg}]^T.$$

A sufficient condition then for the satisfaction of these three constraints is the single constraint: $\|T_{zw}(\mathcal{G}, \mathcal{C})\|_\infty < 1$, where

$$w := [w_{gust}, \gamma_{cmd}]^T = [5\ fps, 3\ deg]^T$$

$$z := [\tfrac{\delta_{stab}}{20\ deg}, \tfrac{\delta_{DLC}}{40\ deg}, \tfrac{\alpha}{1.5\ deg}, \tfrac{\alpha_{error}}{c_1 s}, \tfrac{\gamma_{error}}{c_2 s}]^T,$$

and c_1 and c_2 are any finite positive numbers. This constraint also implicitly guarantees internal stability. This constraint later leads to the formulation of the synthesis model.

The choice of optimization parameters was driven by the unusual character of the Direct Lift Control. This is significant because the DLC control power is neither linear in deflection nor proportional to the size of the DLC surfaces. Consequently, we must optimize a metric whose influence on the plant dynamics is linear. Normalized control power was therefore chosen as the optimization parameter in order to adhere to the assumption that the plant dynamics be reasonably modeled by a linear system. In the actual aircraft implementation, the controller will have to include a nonlinear schedule on the DLC deflection in order to achieve the commanded control power. Since the stabilators are a conventional aerodynamic surface, their control power is in fact linear with both deflection and surface area, and any one of several optimization parameters could have been chosen (tail volume, surface area, or absolute control power). To facilitate the comparison, normalized control power was chosen.

To proceed with the problem description we need to define the following terms using flight dynamics conventions:

u	:=		aircraft velocity resolved in the body x-axis
α	:=		angle of attack
q	:=		pitch rate about body y-axis
θ	:=		pitch attitude
C_{X_t}	:=	$\frac{\partial X}{\partial y}$	non-dimensional trimmed force/moment coefficient, where X is lift (L), drag (D) or pitching moment (M)
C_{X_y}	:=	$\frac{\partial X}{\partial y}$	non-dimensional stability derivative, where $y \neq t$ is a nondimensional state or control deflection
Q	:=		dynamic pressure

S	:=		wing reference area
$\bar{c}$	:=		wing mean chord
m	:=		mass
I_{yy}	:=		moment of inertia about the lateral (y) axis
V	:=		aircraft velocity
σ_α	:=	$\frac{\sigma_{w_{turbulence}}}{V}$	turbulence variance
σ_γ	:=		maximum flight path angle command amplitude
l_{stab}	:=	$-\frac{C_{M_{stab}}}{C_{L_{stab}}}$	stabilator lever arm (distance from CG to aero center of stab)

Next, the aircraft stability derivatives were expressed as the sum of their wing/body (*wb*), stabilator (*stab*), and DLC contributions [Ref. 40]:

$$C_{D_{\alpha_{wb}}} := C_{D_\alpha} \tag{5.32}$$

$$C_{L_{\alpha_{wb}}} := C_{L_\alpha} - C_{L_{stab}} \tag{5.33}$$

$$C_{M_{\alpha_{wb}}} := C_{M_\alpha} - l_{stab}\, C_{L_{stab}} \tag{5.34}$$

$$C_{L_{q_{wb}}} := C_{L_q} - 2\, l_{stab}\, C_{L_{stab}} \tag{5.35}$$

$$C_{M_{q_{wb}}} := C_{M_q} + 2\, l_{stab}^2\, C_{L_{stab}} \tag{5.36}$$

$$C_{L_{\dot{\alpha}_{wb}}} := C_{L_{\dot{\alpha}}} + 2\, l_{stab}\, C_{L_{stab}} \tag{5.37}$$

$$C_{M_{\dot{\alpha}_{wb}}} := C_{M_{\dot{\alpha}}} - 2\, l_{stab}^2\, C_{L_{stab}}. \tag{5.38}$$

Note that since the DLC perturbs the nominal flow field, it has no contribution to the A matrix, but instead represents raw control power, influencing only the B matrix.

The linear aerodynamic model of F-14 is derived next, followed by an outline of how it was used to form a synthesis model. The state derivatives of the core aerodynamic plant were $x = [u \quad \alpha \quad q \quad \theta]^T$. Let T be a rotation/scaling matrix:

$$T := QS \begin{bmatrix} -\cos(\alpha_0) & \sin(\alpha_0) & 0 \\ -\sin(\alpha_0) & -\cos(\alpha_0) & 0 \\ 0 & 0 & \bar{c} \end{bmatrix},$$

where α_0 is the trimmed value of α.

Let I_m represent the inertia/ $\dot{\alpha}$ matrix. Then

$$I_m := \begin{bmatrix} mV & & & \\ & mV & & \\ & & I_{yy} & \\ & & & 1 \end{bmatrix} - \begin{bmatrix} \frac{\bar{c}}{2V} T \begin{bmatrix} 0 & 0 & 0 \\ 0 & C_{L_{\dot{\alpha}}} & 0 \\ 0 & C_{M_{\dot{\alpha}}} & 0 \end{bmatrix} & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ \begin{matrix} 0 & 0 & 0 \end{matrix} & 0 \end{bmatrix}.$$

Now, the state matrices of the plant have the following form:

$$A_{aero} = A_{aero_0} + A_{aero_1}\zeta_1 + A_{aero_2}\zeta_2$$

$$= I_m^{-1} \left[\left(T \begin{bmatrix} 2C_{D_t} + C_{D_u}V^{-1} & C_{D_{\alpha_{wb}}} - C_{L_t} & 0 \\ 2C_{L_t} + C_{L_u}V^{-1} & C_{L_{\alpha_{wb}}} + C_{D_t} & C_{L_{q_{wb}}} \frac{\bar{c}}{2V} \\ 2C_{M_t} + C_{M_u}V^{-1} & C_{M_{\alpha_{wb}}} & C_{M_{q_{wb}}} \frac{\bar{c}}{2V} \end{bmatrix} + \begin{bmatrix} 0 & 0 & -mV\sin(\alpha_0) \\ 0 & 0 & mV\cos(\alpha_0) \\ 0 & 0 & 0 \end{bmatrix} \right) \begin{matrix} -mg\cos(\theta_0) \\ -mg\sin(\theta_0) \\ 0 \end{matrix} \\ \begin{matrix} 0 & 0 & 1 \end{matrix} \quad 0 \end{bmatrix}$$

$$+ I_m^{-1} \begin{bmatrix} C_{L_{stab}} T \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 2\frac{\bar{c}l_{stab}}{2V} \\ 0 & l_{stab} & 2\frac{\bar{c}l_{stab}^2}{2V} \end{bmatrix} & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ \begin{matrix} 0 & 0 & 0 \end{matrix} & 0 \end{bmatrix} \zeta_1 + C_{L_{DLC}} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \zeta_2 \tag{5.39}$$

$$B_{aero} = B_{aero_0} + B_{aero_1}\zeta_1 + B_{aero_2}\zeta_2$$

$$= \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} + I_m^{-1} \begin{bmatrix} C_{L_{stab}} T \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ -l_{stab} & 0 \end{bmatrix} & \\ \begin{matrix} 0 & & \end{matrix} & 0 \end{bmatrix} \zeta_1 +$$

$$I_m^{-1} \begin{bmatrix} C_{L_{DLC}} T \begin{bmatrix} 0 & C_{D_{DLC}}/C_{L_{DLC}} \\ 0 & 1 \\ 0 & C_{M_{DLC}}/C_{L_{DLC}} \end{bmatrix} \\ \begin{matrix} 0 & 0 \end{matrix} \end{bmatrix} \zeta_2. \tag{5.40}$$

Note that these matrices include dynamic coupling and gravity terms, as well as the aerodynamic forces. This model was verified by comparison with the nominal

state matrices ($\zeta = [1,\ 1]^T$) obtained from the linearization of the nonlinear model described in Chapter IV.

Using the aerodynamic plant matrices above, a synthesis model $\mathcal{G}$ was formed:

$$\mathcal{G} = \begin{cases} \dot{x} &= Ax + B_1 w + B_2 u \\ z &= Cx + D_2 w, \end{cases}$$

with w and z defined above, and and

$$A = \begin{bmatrix} & A_{aero_0} & & \begin{matrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{matrix} \\ \begin{matrix} 0 & 1 & 0 & 0 \end{matrix} & & & \begin{matrix} 0 & 0 \end{matrix} \\ \begin{matrix} 0 & -1 & 0 & 1 \end{matrix} & & & \begin{matrix} 0 & 0 \end{matrix} \end{bmatrix} + \begin{bmatrix} & A_{aero_1} & & \begin{matrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{matrix} \\ \begin{matrix} 0 & 0 & 0 & 0 \end{matrix} & & & \begin{matrix} 0 & 0 \end{matrix} \\ \begin{matrix} 0 & 0 & 0 & 0 \end{matrix} & & & \begin{matrix} 0 & 0 \end{matrix} \end{bmatrix} \zeta_1, \tag{5.41}$$

$$B_1 = \begin{bmatrix} \sigma_\alpha A_{aero_0}(:,2) & \begin{matrix} 0 \\ 0 \\ 0 \\ 0 \end{matrix} \\ \begin{matrix} 0 \\ 0 \end{matrix} & \begin{matrix} 0 \\ \sigma_\gamma \end{matrix} \end{bmatrix} + \begin{bmatrix} \sigma_\alpha A_{aero_0}(:,2) & \begin{matrix} 0 \\ 0 \\ 0 \\ 0 \end{matrix} \\ \begin{matrix} 0 \\ 0 \end{matrix} & \begin{matrix} 0 \\ 0 \end{matrix} \end{bmatrix} \zeta_1, \tag{5.42}$$

$$B_2 = \begin{bmatrix} B_{aero_1} \\ \begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix} \end{bmatrix} \zeta_1 + \begin{bmatrix} B_{aero_2} \\ \begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix} \end{bmatrix} \zeta_2, \tag{5.43}$$

$$C = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{\sigma_{\alpha_{out}}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.00001 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.00001 \end{bmatrix} \tag{5.44}$$

$$D_2 = \begin{bmatrix} \frac{1}{u_{max_1}} & 0 \\ 0 & \frac{1}{u_{max_2}} \\ 0 & 0 \\ 0 & 0 \\ 0 & \end{bmatrix}. \tag{5.45}$$

Note, the second column of A shows up in B_1 because the plant sees a sharp edged perturbation in the vertical airmass (δw) as a perturbation in angle of attack ($\delta\alpha = \delta w/V$). (Here $A(:,2)$ represents the second column of A).

Note that the elements of w were scaled by σ_i to achieve $\|w\|_2 \leq 1$. As a result, the synthesis model was scaled to ensure that any state-feedback controller $\mathcal{C} = K$ satisfying $\|T_{zw}(\mathcal{G}, \mathcal{C})\|_\infty < 1$ also satisfies each of the specified design requirements. The scaling of the integral errors was chosen to be small (1×10^{-5}) in order to limit the conservatism of the $\mathcal{H}_\infty$ constraint. Adjusting these values principally influences the time constant of the washouts.

Figure 5.7 depicts the optimization history for this problem. The top graph depicts the progress of the controller margin with each iteration. The second graph shows the progress of the total cost with each iteration The lower graph depicts the progress of the plant parameters. Three optimization runs are shown, initialized at values of $\zeta_0 = [10, \quad 10]^T, [0.12, \quad 50]^T$, and $[1, \quad 20]^T$. The plots were shifted horizontally so as to terminate at the same iteration count.

This example has a number of interesting features. First of all, note that the final output value was independent of the initialization conditions. While this by no means proves the existence of a unique minima in the feasible set, it is an encouraging result. Secondly, unlike the single variable problem, which was known to be quasi-convex, the controller margin does not decrease monotonically with each iteration. Finally, during the progress of the optimization, the stab control power actually falls below its convergence value. What is significant, however, is that the total cost continues to decrease monotonically as the stab control power overshoots its optimal value and then corrects. About a dozen permutations of this problem were run with various weighting vectors and differing performance specifications. The following identically significant behaviors were observed: 1) the output values were always independent of the initialization vector ζ_0; and 2) the total cost always decreased monotonically.

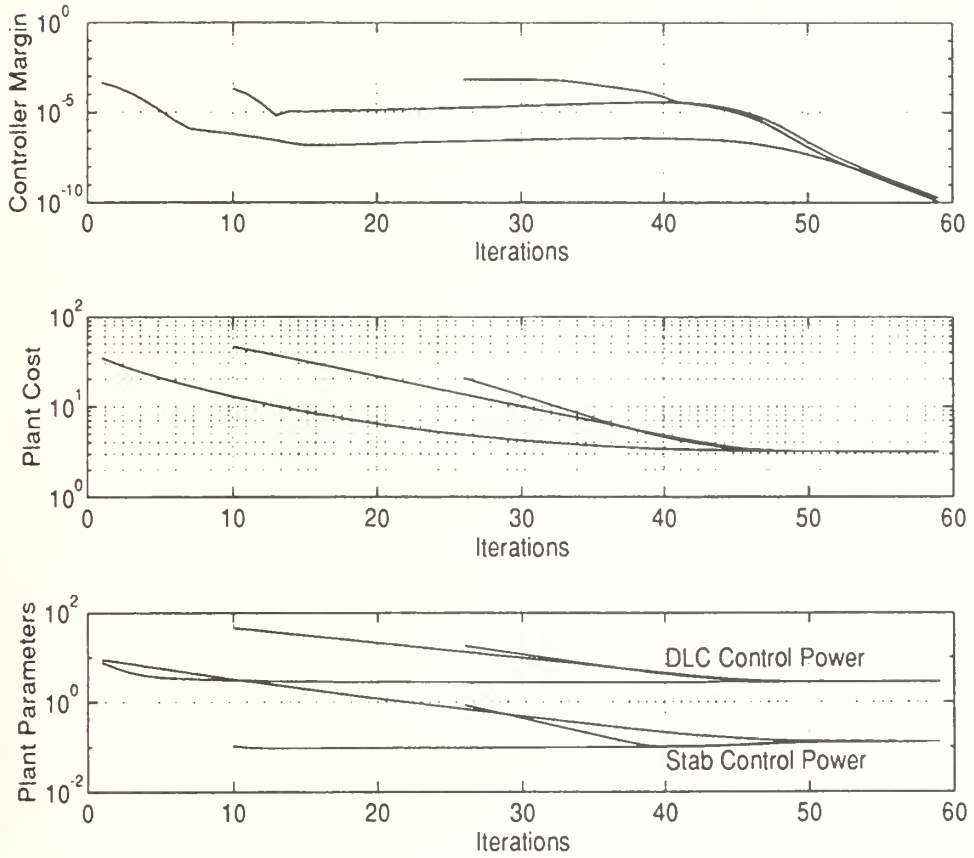

Figure 5.7: Optimization History for Longitudinal F-14 Problem with Multiple Initial Conditions (Example Five)

Another issue of significant importance is actuator activity. It might be expected that the optimization routines minimized control power at the expense of unacceptable increases in actuator activity. This turned out not to be the case, since actuator deflection was constrained as a part of $\mathcal{H}_\infty$ problem formulation. Figure 5.8 shows the broken-loop control responses for the stab and DLC actuators. With cross-over frequencies in the range of one to three rad/s, the actuator demands for this problem are about one order of magnitude less than modern flight control actuator capacity.

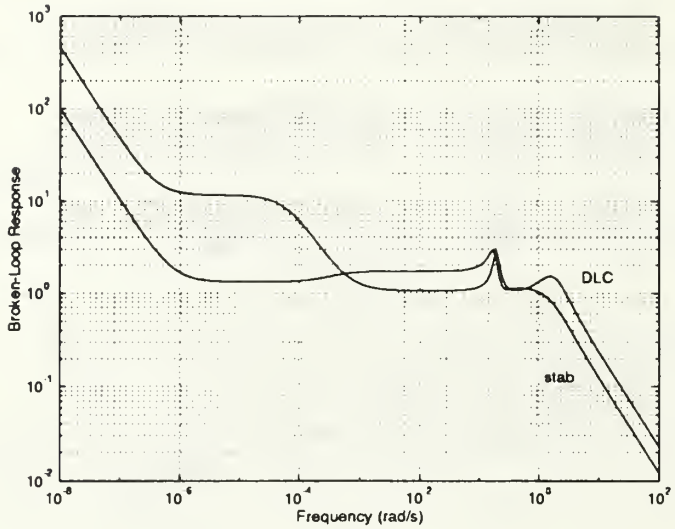

Figure 5.8: Broken-Loop Control Response for Longitudinal F-14 Problem (Example Five)

(2) A Lyapunov Approach

The second approach using the closed-loop formulation was to consider the closed-loop transfer function from the command reference signal to each control as an output, and explicitly limit the steady state value. Let K_i be the row of the state-feedback gain matrix corresponding to the control input u_i, and let $\eta_i := \frac{u_{max_i}}{r_{thres}}$. The closed-loop transfer function from w to u_i is:

$$T_{u_i w}(s) = K_i \left(sI - (A + B_2 K)\right)^{-1} B_1. \tag{5.46}$$

Enforcing

$$T_{u_i w}(0) < \eta_i$$

yields:

$$\eta_i + K_i (A + B_2 K)^{-1} B_1 > 0. \tag{5.47}$$

Note that since η_i is scalar, this is equivalent with the expression:

$$\det \left(\eta_i + K_i (A + B_2 K)^{-1} B_1\right) > 0. \tag{5.48}$$

Recall Schur's determinantal formula [Ref. 41], that given: $M = \begin{bmatrix} F & G \\ H & J \end{bmatrix}$ and $\det(F) \neq 0$:

$$\det(M) = \det(F) \det\left(J - HF^{-1}G\right). \tag{5.49}$$

Consequently,

$$\begin{aligned} \det \begin{bmatrix} -(A + B_2K) & B_1 \\ K_i & \eta_i \end{bmatrix} &= \det\left(\eta_i + K_i(A + B_2K)^{-1}B_1\right) \det\left(-(A + B_2K)\right) \\ &= \det\left(-(A + B_2K) - \frac{B_1K_i}{\eta_i}\right) \det\left(\eta_i\right). \end{aligned}$$

Equation 5.48 is consequently equivalent with:

$$\frac{\det(\eta_i)}{\det\left(-(A + B_2K)\right)} \det\left(-(A + B_2K) - \frac{B_1K_i}{\eta_i}\right) > 0. \tag{5.50}$$

Note that $\det(\eta_i) > 0$. Furthermore, assume $(A + B_2K)$ is stable, which implies that $\det\left(-(A + B_2K)\right) > 0$. The constraint can now be expressed as:

$$\det\left(-(A + B_2K) - \frac{B_1K_i}{\eta_i}\right) > 0. \tag{5.51}$$

Note that $\left(A + B_2K + \frac{B_1K_i}{\eta_i}\right)$ stable is a sufficient (but not necessary) condition for satisfaction of this scalar inequality. If we again use the controller parameterization $K = WY^{-1}$, this can be expressed by Lyapunov's equation as the LMI:

$$AY + YA^T + B_2W + W^TB_2^T + \frac{B_1W_i + W_i^TB_1^T}{\eta_i} < 0. \tag{5.52}$$

An example problem using this formulation were not completed, however, the codes supporting this approach are included in Appendix D.

b. Dynamic Maneuverability Requirements: Open-Loop Formulation

In this section we propose a methodology for inclusion of open-loop maneuverability constraints in a strictly open-loop formulation. As discussed earlier,

such constraints may include maximum pitch rate, roll rate or yaw rate requirement with all the control surfaces set at their maximum deflections. It turns out that these constraints can be derived using the open loop linear plant matrices such as A_{aero} and B_{aero} introduced in the previous section.

Assume that the scalar output $z = r$ is also represented by one of the states of the system, x_i. With all other states zero, the maximum steady state value of x_i can be expressed by:

$$0 = A_{(i,i)}x_i + B_{2_{(i,:)}}u_{max}, \tag{5.53}$$

where $A_{(i,i)}$ denotes the i-th diagonal element of A, and $B_{2_{(i,:)}}$ denotes the corresponding row of B_2. Note that $A_{(i,i)}$ must be negative in order to represent a stable equilibrium condition. The open-loop constraint can now be posed as the inequality:

$$A_{(i,i)}r_{thres} + B_{2_{(i,:)}}u_{max} < 0. \tag{5.54}$$

This is now affine in our plant matrices, and the plant optimization constraint can be posed as the LMI:

$$-A_{(i,i)}(\zeta)z_{thres} - B_{2_{(i,:)}}(\zeta)u_{max} > 0. \tag{5.55}$$

Note that the assumption can be relaxed that z be represented by a system state. This is because if $z = Cx$, then a similarity transform S exists which can make z a state $\hat{x}_i$ of an equivalent system:

$$\dot{\hat{x}} = \hat{A}\hat{x} + \hat{B}_2 u : \quad z = \hat{x}_i. \tag{5.56}$$

Furthermore, if A and B_2 are affine in ζ, then $\hat{A}(\zeta) = SA(\zeta)S^{-1}$ and $\hat{B}_2(\zeta) = SB_2(\zeta)$ are affine in ζ.

Constraint 5.55 can therefore be posed jointly with any of the other proposed constraints, and the plant/controller optimization problem solved as before.

Because this constraint is truly open-loop, and only applies to the plant optimization phase of the routine, it can be omitted from the constraints imposed during the closed-loop controller optimization as with constraint 5.29.

Example Six– Optimization of Multiple Surfaces with Open and Closed-Loop Performance Requirements

Let's revisit the F-14 longitudinal example. This time consider the additional open-loop maneuverability requirement that adequate control power exist to generate at least 0.5 rad/sec of positive pitch rate with the stabilator deflected 20 degrees from the trimmed value, and the DLC neutral. Consequently, $stab_{max} = \frac{20}{57.3}$ radians and equation 5.55 now becomes:

$$- A_{aero_{(3,3)}}(\zeta)(0.2rad) - B_{aero_{(3,1)}}(\zeta)\frac{20}{57.3} > 0. \tag{5.57}$$

This LMI was included in the plant optimization phase of the plant/controller algorithm. Figure 5.9 depicts the progress of the algorithm. The addition of this open-loop maneuverability constraint resulted in an increase in the required size of the stabilator from 0.12 to 0.34 of its normalized value. The DLC size predictably remained the same as in Example Six. The results shown in Figure 5.9 indicate that the closed-loop $\mathcal{H}_\infty$ performance constraint was responsible for sizing the DLC, while the open-loop maneuverability constraint was responsible for appropriately sizing the stabilators. The abrupt change in the controller margin is for the identical reason as that explained in Example Four.

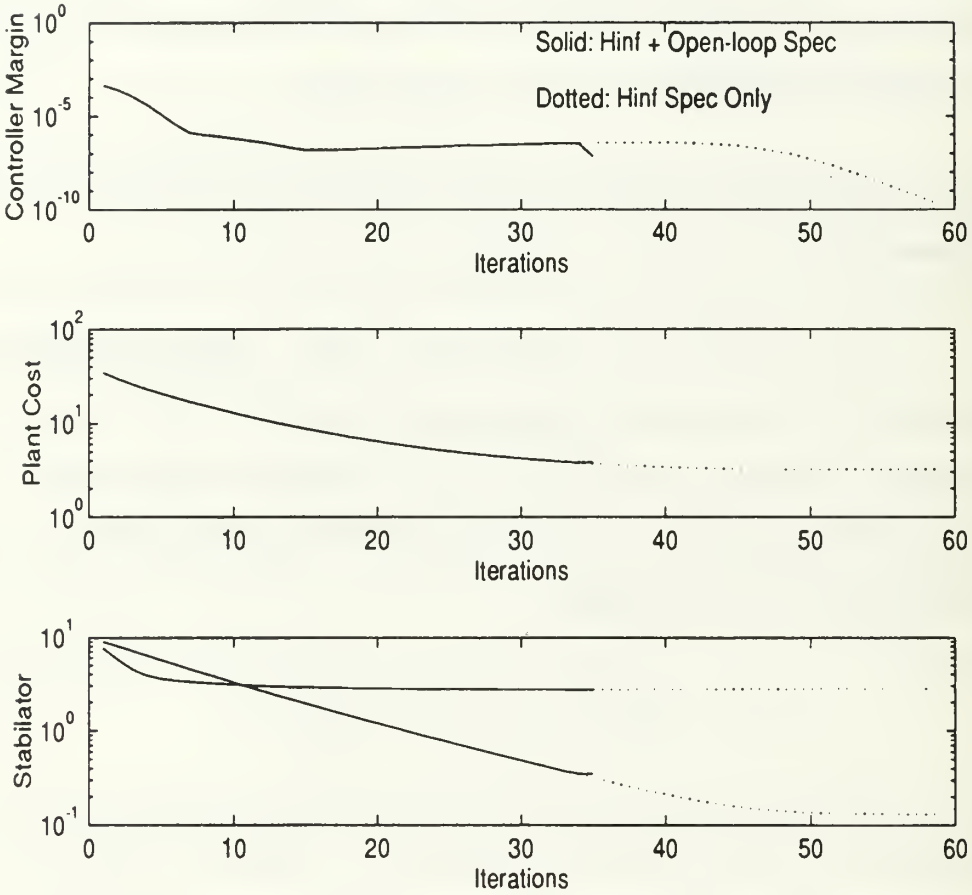

Figure 5.9: Optimization History for F-14 Problem with Open-Loop Constraint (Example Six)

Example Seven– Optimization of Multiple Surfaces with Open and Closed-Loop Performance Requirements and Directed Thrust

The objective of this example is to demonstrate that more unconventional control effectors can be easily incorporated into the proposed methodology. Therefore, in addition to the aerodynamic control surfaces, provision was made in the model for the deflection of the engine exhaust to provide additional moment (directed thrust). In this simplified model the thrust is set to the trimmed value of 13,118 *lbf*,

and ancillary nonlinear effects of thrust vectoring, such as entrainment, are ignored. The commanded thrust deflection is set to be equal to the stabilator deflection, with similar actuation rates, such that $\delta_{thrust} = \delta_{stab}$. Consequently, the controls problem remains a two-input problem.

We first discuss incorporation of the directed thrust into the synthesis model. Define the following parameters:

$$\begin{array}{rcl} T_0 & := & \text{trimmed thrust value } (lbf) \\ l_\epsilon & := & \text{engine lever arm } (ft) \\ \theta_\epsilon & := & \text{thrust deflection angle } (rad), \text{ relative to body axis} \\ \theta_{\epsilon_0} & := & \text{nominal installed thrust deflection angle } (rad). \end{array}$$

Using above notation and the notation of the previous section, the nonlinear directed thrust contribution to the state derivatives can be expressed as:

$$\begin{bmatrix} \dot{U} \\ \dot{\alpha} \\ \dot{Q} \\ \dot{\Theta} \end{bmatrix}_{thrust} = T_0 \begin{bmatrix} \frac{1}{m}\cos(\theta_\epsilon) \\ -\frac{1}{mV}\sin(\theta_\epsilon) \\ -I_{yy}^{-1} l_\epsilon \sin(\theta_\epsilon) \\ 0 \end{bmatrix}. \tag{5.58}$$

Linearizing equation 5.58 about a nominal flight condition results in the following expression:

$$\begin{bmatrix} \dot{u} \\ \dot{\alpha} \\ \dot{q} \\ \dot{\theta} \end{bmatrix}_{thrust} = T_0 \begin{bmatrix} -\frac{1}{m}\sin(\theta_{\epsilon_0}) \\ -\frac{1}{mV}\cos(\theta_{\epsilon_0}) \\ -I_{yy}^{-1} l_\epsilon \cos(\theta_{\epsilon_0}) \\ 0 \end{bmatrix} (\theta_\epsilon - \theta_{\epsilon_0}). \tag{5.59}$$

Incorporating directed thrust in the aircraft dynamics results in the new expression for B_{aero} defined in equation 5.40:

$$\begin{aligned} B_{aero} &= B_{aero_0} + B_{aero_1}\zeta_1 + B_{aero_2}\zeta_2 \\ &= \begin{bmatrix} -\frac{1}{m}T_0\sin(\theta_{\epsilon_0}) & 0 \\ -\frac{1}{mV}T_0\cos(\theta_{\epsilon_0}) & 0 \\ -\frac{1}{I_{yy}}T_0 l_\epsilon \cos(\theta_{\epsilon_0}) & 0 \\ 0 & 0 \end{bmatrix} + I_m^{-1} \begin{bmatrix} C_{L_{stab}} T \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ -l_{stab} & 0 \end{bmatrix} \\ 0 \qquad\qquad 0 \end{bmatrix} \zeta_1 + \end{aligned}$$

$$I_m^{-1}\left[\begin{array}{c} C_{L_{DLC}}T\begin{bmatrix} 0 & C_{D_{DLC}}/C_{L_{DLC}} \\ 0 & 1 \\ 0 & C_{M_{DLC}}/C_{L_{DLC}} \end{bmatrix} \\ \begin{matrix} 0 & 0 \end{matrix} \end{array}\right]\zeta_2. \tag{5.60}$$

Note, whereas B_{aero_0} had previously been zero, it has now includes the directed thrust contribution.

The open-loop maneuverability specification given by equation 5.57 retains the same form. When expanded, this inequality is now:

$$-A_{aero_{(3,3)}}(\zeta)q_{max} - B_{aero_{(3,1)}}(\zeta)\delta_{stab_{max}} - \frac{l_e T_0 \cos(\theta_{e_0})}{I_{yy}}(\theta_e - \theta_{e_0}) > 0. \tag{5.61}$$

The expectation was that significantly less stabilator control power would be required due to the ability of the directed thrust to provide large moments.

As can be seen from Figure 5.10, the methodology determined that essentially no stabilator control power was required in order to meet the specifications, and a controller was found which satisfied the requirements using directed thrust and DLC alone. Note that the availability of the directed thrust did not decrease the amount of DLC control power necessary to meet the requirements. This was to be expected since the influences of these two control inputs on the plant dynamics are nearly orthogonal.

This example demonstrated that that diverse control inputs, such as directed thrust *can* be handled by the methodology we've proposed. However, additional constraints will have to be imposed to guarantee a satisfactory solution for flight conditions where thrust setting is not constant, such as terminal area flight.

In this section we have shown that the proposed methodology can easily accomodate various static and dynamic maneuverability requirements. In particular, it was demonstrated that these requirements can be formulated as LMI's and

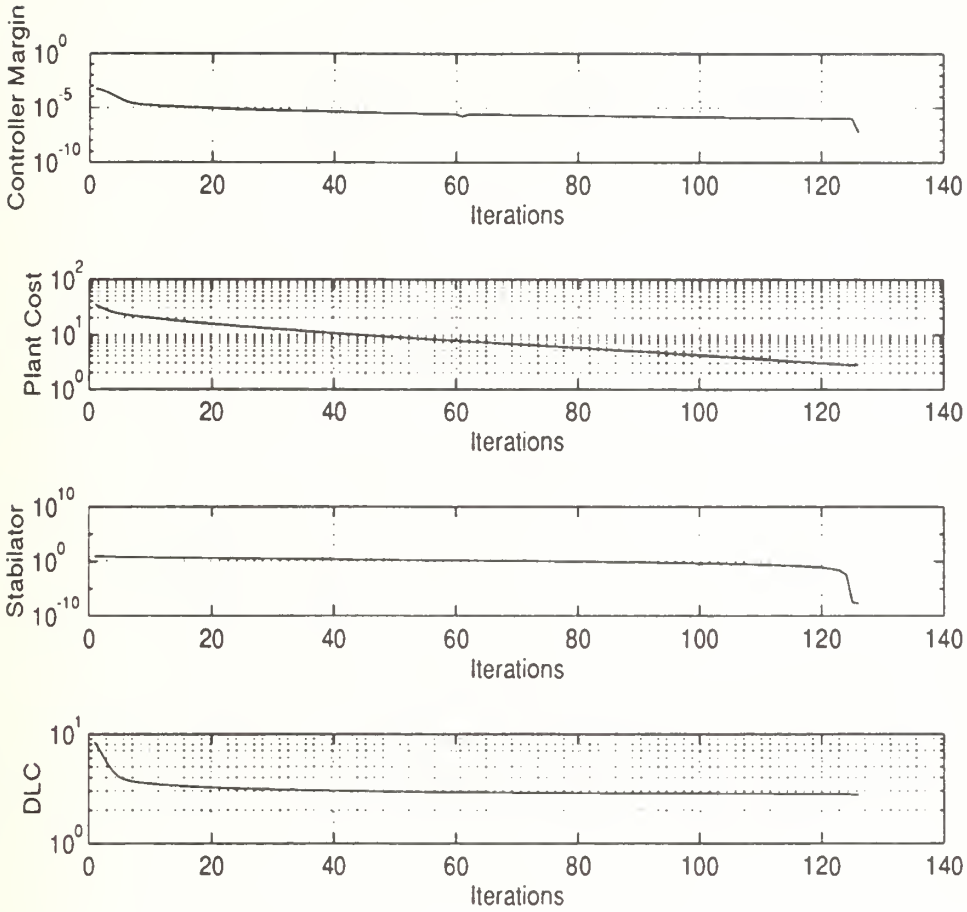

Figure 5.10: Optimization History for F-14 Problem with Directed Thrust (Example Seven)

included in the plant/controller optimization algorithm together with the closed-loop performance and stability constraints.

F. ACCOMODATING MODEL UNCERTAINTY

The hazards associated with under-designing the control power are extreme, and yet the answers determined by the methods above presume perfect knowledge of the linear plant. As a consequence, sound engineering practice would demand that provision was made in the design process to ensure that inaccuracies in the

linear synthesis model would not be responsible for the gross under-design of the control surfaces. If one's confidence in the aerodynamic derivatives was within ten percent, then the temptation might be to simply pad the tail size by ten percent. This methodology would fail to recognize that it is the dynamics of the whole vehicle which are uncertain, and might result in sub-optimal allocation of the control power to cope with the other uncertainties in the knowledge of the plant. Because of the small gain theorem (Theorem 3.7), the $\mathcal{H}_\infty$ methodology of section C was easily extensible to the problem of providing robust control, and optimization of the vehicle control power which would guarantee both robust stability and fulfillment of the performance objectives.

This proves to be an interesting problem, because the formulation of the synthesis model for plant optimization with a robust stability constraint is significantly different from the $\mathcal{H}_\infty$ constraint posed above. Consider again the uncertainty model for longitudinal aircraft dynamics presented in Chapter IV, in which parametric uncertainty was modeled by uncertainties on the total lift, drag and moment generated by purely aerodynamic forces. Figure 4.1 depicted the inputs and outputs of the uncertainty block. Consider the linearization of that nonlinear plant, replacing F_{grav} and F_{dyn} with A_{grav} and A_{dyn}. Furthermore, let $\tilde{A}_{aero}$ represent that the aerodynamic influences such that A_{aero} introduced in equation 5.39 has been decomposed into its aerodynamic, gravity and dynamic coupling contributions, such that $A_{aero} = A_{grav} + A_{dyn} + R_{wb}\tilde{A}_{aero}$.

The uncertainty inputs and outputs to the linear system can then be represented as depicted in Figure 5.11. The system depicted by Figure 5.11 can be posed in a state-space representation:

$$\mathcal{G}_{\Delta} := \left\{ \begin{array}{rcl} \dot{x} & = & (A_{aero}(\zeta) + B_2(\zeta)K)x + R_{wb}w_\delta \\ z_\delta & = & (\tilde{A}_{aero}(\zeta) + B_2(\zeta)K)x \end{array} \right. . \tag{5.62}$$

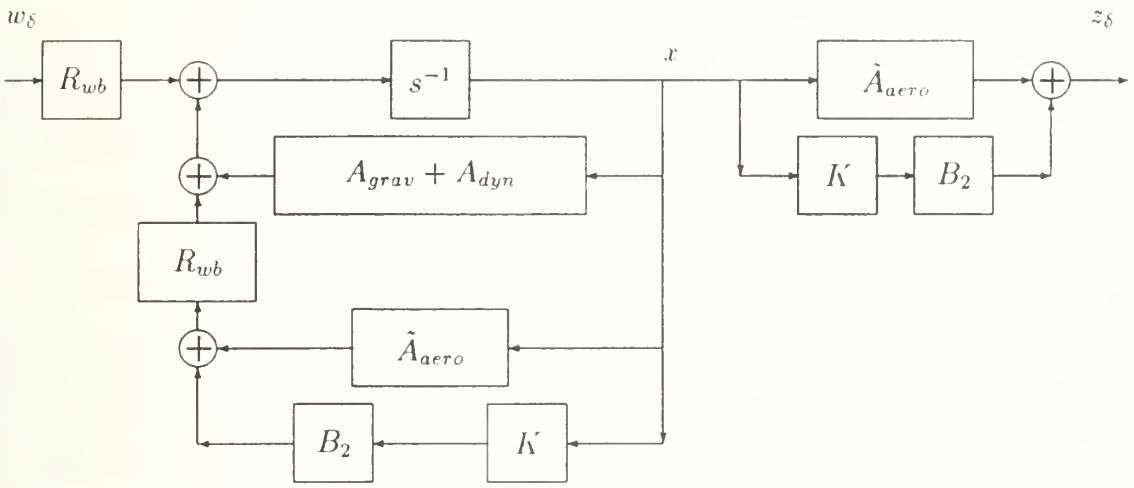

Figure 5.11: Linearized Uncertainty Model

By the Small Gain Theorem, the closed-loop system will be stable for all $\{\Delta : \|\Delta\|_\infty < 1\}$, if and only if $\|T_{z_\delta w_\delta}\|_\infty < 1$. This constraint can be expressed by the LMI:

$$R_\Delta(\xi,\zeta) := \begin{bmatrix} \begin{pmatrix} A_{aero}(\zeta)Y + YA_{aero}(\zeta)' + \\ B_2(\zeta)W + W'B_2(\zeta)' + R_{wb}R_{wb}' \end{pmatrix} & (\tilde{A}_{aero}(\zeta)Y + B_2(\zeta)W)' \\ (\tilde{A}_{aero}(\zeta)Y + B_2(\zeta)W) & -I \end{bmatrix} < 0. \tag{5.63}$$

The problem is different from those encountered previously in that this single matrix inequality is affine in both the controller parameters $(W(\xi), Y(\xi))$ and the plant parameters ζ. This LMI can consequently be used for both phases of the optimization procedure.

Time precluded the completion of a design example using this formulation, though the scripts which support this problem are included in Appendix D.

G. GENERAL COMMENTS

This section presents several diverse comments on the methodology presented above.

1. Interpreting the Results

What does the output of this methodology mean? Two mathematical issues are significant. First of all, though each of the two phases of the method are affine problems, and despite considerable effort, we have been unable to mathematically establish that the feasible set for the joint controller/ plant optimization problem is convex. Consequently, absent a proof of convexity of the feasible set, the existence of lower cost solutions can not be discounted. Secondly, a feasible solution to the appropriate LMI is both necessary and sufficient for the existence of a state-feedback controller satisfying either the $\mathcal{H}_\infty$ constraint or the pole placement specification. When more than one $\mathcal{H}_\infty$ constraint is jointly imposed, or the $\mathcal{H}_\infty$ constraint is imposed jointly with another constraint, such as pole placement, then a feasible solution to the joint LMI is clearly a sufficient condition for a feasible controller. It is no longer, however, a necessary condition [Ref. 25]. Again, absent a proof of necessity, controllers might exist which would permit further reduction of J. The output $J = c^T \zeta_{opt}$ is consequently an upper-bound to the optimum value of J.

2. The Example Problems

It is important to mention that the first several example problems could have been solved by easier means. These examples were not intended to suggest that our methodology should be used to solve these types of problems, but rather to demonstrate the methodology on small easily visualized problems. The answers in fact confirmed our intuition into these simple problems. The power of the methodology is its direct applicability to much more complex problems with multiple variables and

joint constraints, for which no other direct method exists.

3. Limitations of the Methodology

It is important to identify the limitations of the methodology. The above examples demonstrate the viability of the method, and its flexibility to simultaneously adjust to diverse specifications. There were however two assumptions which implicitly limit its application:

1. The linear model faithfully represents the dynamic character of the plant
2. The plant could reasonably be expressed as an affine function of the optimization parameters.

Neither of these assumptions was considered to be significantly limiting, as there are a wide range of examples for which they are both reasonable. It is worth noting for the reader one example for which an aircraft would not be affinely dependent upon a control size. In the case of a closely-coupled canard, the affine assumption would not be reasonable, as the size of such a canard can have a dramatic effect on the influence coefficients of the surfaces located in its wake.

Furthermore, there is one other significant explicit limitation— the existence of a feasible controller. Several example problems were attempted for which *no* feasible controller could be found. This was most common when multiple joint specifications were imposed. The methodology cannot guarantee *a priori* the existence of a controller in the presence of conflicting, or mutually incompatible, performance criteria.

4. Other Applications

The focus of both the discussion and the examples has been the optimization of control power, i.e. those items whose principal influence manifests itself in the B matrix of the state-space representation. Furthermore, the examples have each

suggested that the size of surfaces might be the principal figure of interest. The application of the methodology is not, however, restricted to those features of a vehicle which represent control power. Any feature whose contribution to the dynamic system can be posed in an affine manner can be included in the vector of plant parameters, and consequently be reflected in the cost function. Strakes and fixed fins would be examples of features which might only have a contribution in the A matrix, but are likewise perfectly suited to the application of this method. Occasionally, destabilizing features such as blisters, antennas or external stores must be appended to a vehicle. The methodology can then be applied to find the *maximum* acceptable size of the feature by including the negative contribution of the feature in $A(\zeta)$, and finding the minimum negative value of the sizing parameter. Finally, recall that in the example problems that the tail volume was defined as the normalized product of the surface area and its distance from the *cg*. The influence of a feature is consequently also affine in its position as well as its size, and so the methodology might instead be applied to optimizing position rather than size for some applications. The method presented in this chapter is consequently broadly applicable to a wide variety of problems not demonstrated here.

H. FUTURE DIRECTIONS

The following subjects present the foundations for future work.

1. Convexity Issues

Though the value of the above methodology does not hinge on the problem being proven to be convex, convexity remains an interesting subject. This is particularly true if the method is to be expanded to other applications. One particularly interesting direction to be pursued is the method's close resemblance to D-K iteration [Ref. 34]. D-K iteration is a method by which robust controllers are designed

by iteratively designing a controller and then performing a similarity transform on selected inputs and outputs to scale for robustness. Though the method has been recognized and used for some time by the controls community , special cases of the problem were just recently proven to be convex [Ref. 42, 43]. The similarity of these two problems tenders some hope that the problem we've posed may yet prove to be convex.

In the event that general convexity can not be demonstrated, perhaps the mathematical nature of the problem can further be refined to enhance our understanding of the result. Can a lower bound be computed? Is the result a local or global minimum? Are there other special conditions for which convexity of the problem can be assured or imposed? For example, if one could determine the smallest convex set which contained the feasible set, then a lower bound could be found by minimizing the cost function over that convex set. A lower bound close to the upper-bound found through our iterative methodology would be of tremendous practical value. These are subjects which each warrant further investigation.

2. Other Convex Performance Constraints

A second limitation to the above approach was its restriction to those convex constraints for which an LMI formulation exists. A number traditional performance metrics are convex problems for which an LMI formulation does not presently exist [Ref. 13], including overshoot, rise-time, settling time, and response bandwidth. The above methodology can and should be expanded to include these types of performance specifications.

I. CONCLUSIONS

While establishing the minimum control power requirements for a aircraft has always been a concern, it is now a core design constraint with the advent of aircraft

whose open-loop dynamics may be unstable. With an unstable platform, the control effectors must now not only provide the capacity for adequate maneuverability, but also provide stabilization and disturbance rejection. For this type of vehicle, if an control actuator encounters a deflection or deflection rate limit in providing maneuverability, it reverts to its natural dynamics in response to any disturbance. Missteps in this arena have been well documented. The loss of the prototype Grippen on its sixth flight was attributed to saturated control rates in an environment where linear aero and flight dynamics prevails [Ref. 44]. To under-design the control power is to court disaster. To compensate for the hazards by over designing the control power is to forfeit all the benefits offered by reduced static stability (low monetary cost, high maneuverability, low drag, low signature, low weight).

Direct numerical methods have been needed in which performance requirements, including actuator rates and deflections, are the inputs to a methodology which then can simultaneously determine both the lowest "cost" configuration and its accompanying controller. This chapter demonstrated that many common flying qualities specifications can be posed as Linear Matrix Inequalities. These included stabilization requirements, disturbance rejection requirements, and static and dynamic maneuverability requirements. Furthermore, a iterative method of optimizing a plant configuration was demonstrated when the performance constraints can be posed as Linear Matrix Inequalities. This work is at a threshold, having demonstrated a viable means for solving a significant subset of control power problems, and suggesting an approach to pursue the larger set of convex performance constraints, which predominate the flying qualities specifications.

J. RECOMMENDATIONS

It is recommended that the following steps be taken to further the contribution of this work:

1. Procure commercial interior point codes when they become available. The point here is to separate the influence of the numerical methods from the application of the engineering formulation. The coding of the interior point algorithm in Appendix B restricts the engineering application of the above formulation in two ways: (1) it is slow due to the choice of MATLAB as the programming language, and (2) the code demonstrated some irregularities, such that there were some problem geometries for which the path of centers was unstable. Consequently some interesting feasible example problems were attempted which the available interior point code would not solve.

2. Extend the general methodology to other performance measures which have convex solutions, but which do not currently have LMI formulations. Many other relevant convex constraints exist, such as command and controller bandwidths, which could also be applied if the method were extended to non-LMI formulations.

3. Continue to pursue attempts to either prove convexity or find associated convex problems which could be used to determine a lower bound.

VI. CONCLUSIONS AND RECOMMENDATIONS

This report has treated the subject of the application of $\mathcal{H}_\infty$ control design and convex optimization to the design of air vehicles and their control systems. Three project areas defined the scope of the research effort:

- An $\mathcal{H}_\infty$ controller design for the F-14 autoland problem, implementing Direct Lift as an active control surface.
- The programming of mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ design tools, and their application to the autoland controller design problem.
- The formulation of the plant/controller optimization problem in a format where an upper-bound can be determined by convex methods.

Each of these efforts were successfully completed, each with their own sets of conclusions.

A. $\mathcal{H}_\infty$ DESIGN EXAMPLE CONCLUSIONS AND RECOMMENDATIONS

The $\mathcal{H}_\infty$ output-feedback synthesis methods were used to determine an autoland controller for a carrier-based F-14 aircraft. Significant results of this efforts were:

1. A methodology was demonstrated for the formulation of robustness analysis models for aeronautical applications. This methodology is appropriate for those problems in which the origin of the parameteric data is flight test rather than wind tunnels.

2. Direct Lift Control was successfully implemented in a MIMO design for the autoland problem.

3. A methodology was demonstrated whereby SISO design specifications were translated in scalar weighting functions on the inputs and outputs of the synthesis model. This included tuning the controller for specified bandwidths of the measurement sensors.

4. A controller was designed which did not cancel the open-loop poles of the plant.

5. The controller performance was validated by nonlinear simulation.

This design example extended the methodology of previous work to the full measurement feedback problem. The methodology is recommended for consideration whenever $\mathcal{H}_\infty$ control design methods are considered for use. Its principal attribute is that it represents a methodology by which a controls designer with limited background in the theoretical aspects of $\mathcal{H}_\infty$ control can still apply the design tools with confidence.

B. MIXED $\mathcal{H}_2$ / $\mathcal{H}_\infty$ CONTROL CONCLUSIONS AND RECOMMENDATIONS

The mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ control phase of the research included both the coding of the necessary design tools for both the continuous and discrete time problems, as well as their application to a design problem similar to the F-14 autoland problem above. Siginifcant results included:

1. Design tools were created which solve the continuous and discrete time mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ controller synthesis problems using elliptical convex optimization methods.

2. The F-14 autoland control problem was used to demonstrate a methodology whereby the continuous $\mathcal{H}_2$ / $\mathcal{H}_\infty$ control design tools could be used in a MIMO controller synthesis problem.

Due to the computational time required to use the mixed tools, it is recommended that they only be applied for design problems in which an $\mathcal{H}_2$ specification was explicitly part of the design requirements.

C. PLANT/CONTROLLER OPTIMIZATION CONCLUSIONS AND RECOMMENDATIONS

Chapter V was devoted to outlining and demonstrating a methodology whereby the optimization of a vehicle's physical configuration could be posed for solution by the methods of convex analysis. Specifically:

1. It was demonstrated that many typical flying qualities requirements can be jointly posed as Linear Matrix Inequalities, including:
   - Stabilization requirements, including pole placement,
   - Disturbance Rejection,
   - Static and Dynamic Maneuverability (Open-loop) Requirements.
2. It was demonstrated that plant dynamics are frequently affinely dependent on some physical attributes of the physical configuration, such as surface size.
3. A methodology was presented for determining an upper-bound to the plant/controller optimization problem. The method is appropriate for problems in which the design specifications can be posed as Linear Matrix Inequalities and the plant dynamics are affinely dependent upon plant parameters. This problem was shown to be quasi-convex for the optimization of single plant parameters. It

was also shown how the method results in the determination of an upper-bound for the cost functions for multiple parameter problems.

4. A methodology was outlined for accommodating modeling uncertainties into the plant/controller optimization problem .

5. Multiple design examples were presented whereby joint performance requirements were applied to the problem of optimizing aircraft control surface configurations.

This represents the principal major contribution of this research effort. It is recommended that the following items be pursued in this area:

1. Commercial interior point codes should be procured when they become available.

2. The general methodology should be extended to other performance measures which have convex solutions, but which do not currently have LMI formulations.

3. The issue of convexity should continue to be explored, including attempts to identify bounding convex sets.

APPENDIX A:ALGORITHMS FOR THE SOLUTION OF THE MIXED $\mathcal{H}_2$ / $\mathcal{H}_\infty$ CONTROLLER SYNTHESIS PROBLEMS

This appendix presents the numerical tools which were coded to solve the mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ control synthesis problem. It includes both the mathematical derivations of various terms, as well as verbatim listings of the final codes. Since no numerical codes were available to solve these problems, a considerable amount of time was devoted to the design of the necessary numerical routines. The optimization problems in this appendix were all solved by the ellipsoidal method.

First, section A presents several simple, but perhaps obscure, relationships which were critical to the derivation of analytical expressions for the gradients of the various matrix functionals. Next, section B presents the codes associated with the continuous time $\mathcal{H}_2$ / $\mathcal{H}_\infty$ problem. Section C then presents the codes associated with the discrete time mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ synthesis problem. Finally, section D presents a short discussion of the validation of the codes.

A. FINDING THE GRADIENTS OF MATRIX FUNCTIONALS

Both the continuous and discrete $\mathcal{H}_2$ / $\mathcal{H}_\infty$ state-feedback problems can be numerically solved by the straightforward application of the ellipsoidal convex optimization algorithm, described in Chapter II. In both cases, the most challenging part was the derivation and coding of analytical expressions for the subgradients of non-differentiable functions. The following relationships are instrumental in the derivations of the subgradients that follow.

1. Derivative of an Eigenvalue of a Symmetric Matrix Functional

Since most of the constraint functionals (and some objective functionals) are of the form $Q(x) < 0$, it is necessary to be able to determine the gradient of $\phi(x) := \lambda_{max}(Q(x))$. From [Ref. 45], given a symmetric matrix functional $Q(x) = Q(x)^T \in \mathbf{R}^{n\times n}$ operating on $x \in \mathbf{R}^s$, and the scalar function $\phi(x) := \lambda_{max}(Q(x))$; the gradient of $\phi(x)$ is g, such that for each element of g:

$$g_i := \frac{\partial \phi(x)}{\partial x} = u^*(x)\frac{\partial Q(x)}{\partial x_i}u(x), \text{ for all } i = 1,\ldots,s \tag{A.1}$$

where $u(x)$ is the eigenvector associated with the maximum eigenvalue of $Q(x)$, and $u^*(x)$ is its conjugate transpose.

2. Derivative of a Matrix Inverse

Matrix inverses occur frequently in various Riccati equations. An expression for their derivatives is consequently required. From [Ref. 46]:

$$d(X^{-1}X) = d(I) = 0 = d(X^{-1})X + X^{-1}dX,$$

hence,

$$d(X^{-1}) = -X^{-1}dXX^{-1}$$

B. CONTINUOUS TIME MIXED $\mathcal{H}_2$ / $\mathcal{H}_\infty$ CONTROLLER DESIGN

As discussed in Chapter III and [Ref. 11], the continuous mixed output-feedback control problem is based upon a convex expression of the state-feedback problem. It can be solved by the solution of the filtering Riccati equation, and the construction of an auxiliary plant for which a state-feedback controller is designed. This section consequently develops and lists the state-feedback controller design tools which can either be used independently or called by the output-feedback synthesis function. The measurement-feedback controller synthesis tools then follow.

1. Numerical Solution of the Continuous Time State-Feedback Problem

The generalized mixed continuous $\mathcal{H}_2$ / $\mathcal{H}_\infty$ state-feedback problem is posed above in Theorem 3.4 as:

Minimize:

$$f_i(M(W,Y)) := f_i\left\{(C_0Y + D_{02}W)Y^{-1}(C_0Y + D_{02}W)^T\right\}, \tag{A.2}$$

Subject to: $Y > 0$, and $R(W,Y) < 0$,

where

$$R(W,Y) := AY + YA^T + B_2W + W^TB_2^T + B_1B_1^T + (C_1Y + D_{12}W)(C_1Y + D_{12}W)^T, \tag{A.3}$$

and f_1 is the trace, f_2 is the maximum eigenvalue, or f_3 is the maximum diagonal element. The $\mathcal{H}_\infty$ constraint is assumed to be $\gamma < 1$, and w and z are assumed to have been scaled in order that the constraint $\| T_{z_1w} \|_\infty < 1$ is feasible. The codes which follow also required that w and z have been appropriately scaled.

For the state-feedback problem, the controller is a constant gain matrix K which has been parameterized as $K = WY^{-1}$. Since $Y = Y^T \in \mathbf{R}^{n\times n}$ is symmetric, the problem is a search over the vector space $\mathbf{R}^s$, where $s = \frac{n(n+1)}{2} + nq$. For simplicity and reduced computational expense, Y and W were chosen to be affine functions of x: $Y = \sum_{i=1}^{s} x_iY_i$, and $W = \sum_{i=1}^{s} x_iW_i$. The easiest possible mapping from $x \rightarrow (W,Y)$ was to assign each x_i to a single location (or pair of locations in the case of the off-diagonal elements of Y). The basis matrices Y_i and W_i therefore orthonormal with a single unit one (or pair of ones). For $i = 1$ to n, Y_i had a single unit value in the corresponding diagonal position, and $W_i = 0$. For $i = n + 1$ to $\frac{n(n+1)}{2}$, each Y_i had a pair of symmetrical ones placed by counting down each sub-diagonal, from the first sub-diagonal out to the corners. For $i = \frac{n(n+1)}{2}$ to $\frac{n(n+1)}{2} + nq$, each W_i was all zero

with the exception of a single one, counting down the columns from left to right. For this range $Y_i = 0$. This structure is significant in that when Y_i and W_i show up in the gradient expressions, the multiplication was not required, but the appropriate row(s) or column(s) simply selected out.

An ellipsoidal algorithm was coded in MATLAB following the method of Chapter II and using the following structure:

1. Determine if the $\mathcal{H}_\infty$ problem is feasible, and determine a particular feasible controller Kp. This was done by solving the $\mathcal{H}_\infty$ synthesis equation using the Riccati solvers in the μ-tools toolbox [Ref. 24].

2. Determine an initial feasible solution Y_p and $W_p = K_p Y_p$ from a solution to the $\mathcal{H}_\infty$ analysis equation (3.2).

3. Initialize the search at $x = [0, \ldots, 0]^T$, with a very large ellipsoid about x.

4. From x, determine Y and W as perturbations of the particular solution (W_p, Y_p).

5. Evaluate the constraint $\phi_1(x) = \lambda_{max}(-Y) < 0$. If $\phi_1(x) \geq 0$, then use the eigenvector associated with the maximum eigenvalue to calculate the subgradient: $g_1 = \frac{\partial \phi_1(x)}{\partial x}$. Update the ellipsoid and x and return to step 4.

6. Evaluate the constraint $\phi_2(x) = \lambda_{max}(R(W, Y)) < 0$. If $\phi_2(x) \geq 0$, then use the eigenvector associated with the maximum eigenvalue to calculate the subgradient: $g_2 = \frac{\partial \phi_2(x)}{\partial x}$. Update the ellipsoid and x and return to step 4.

7. With both constraint functionals satisfied, evaluate the objective function $\psi(x) = f(M(W, Y))$ and its subgradient: $g_3 = \frac{\partial \psi(x)}{\partial x}$. Update the ellipsoid, x, and the upper and lower bounds on the estimate of the optimum cost.

8. Return to step 4 unless the termination criteria is satisfied.

9. Calculate the (sub)optimal controller $K = WY^{-1}$.

10. Exit.

Two significant computational issues were involved in actually coding the above routine: 1) how to test for positive definiteness; and 2) how to calculate the maximum eigenvalue and its corresponding eigenvector (both for the subgradients and the f_2 generalized cost). The first was important because both of the above constraint functions required a true/false assessment of the negative definiteness of $-Y$ and R. Two methods were considered: Cholesky factorization, and the computation of the maximum eigenvalue. If a Cholesky factorization of Y or $-R$ could be computed, then that expression was positive definite, and the procedure could move on to the next step. If the factorization failed, then the maximum eigenvalue and its corresponding eigenvector would have to be calculated for the subgradient calculation. The advantage of a Cholesky test, is that it is a very efficient calculation, which, if successful, would avoid the comparatively expensive eigen-problem for that step. As for the determination of the maximum eigenvalue, two choices were apparent. Clearly one choice would be to invoke MATLAB's `eig` function and solve the whole eigen-problem. The second choice would be to use inverse iteration to isolate just the maximum eigenvalue and it's vector. The `eig` function was chosen because the MATLAB implementation is native to the MATLAB core program, and takes advantage of symmetry for problems such as this. It was consequently faster to do the entire eigen-problem by `eig` than run a line-compiled subroutine, though the later required fewer flops. Similar results were encountered in choosing a method for the definiteness test. Though a version of `h2infsyn` successfully ran using a Cholesky test, it was no faster than the eigen test because of `eig`'s implementation directly in MATLAB. Furthermore, the Cholesky dependent variant was not as reliable when

large problems were run. Consequently, the reader will note that `eig` was chosen in resolving both issues. These choices should be reevaluated in the event that these codes are translated into either FORTRAN or 'C'.

The gradient expressions are derived below for the each of the $i = 1$ to n elements:

$$\begin{aligned}
g_{1_i} &= \frac{\partial \phi_1(x)}{\partial x_i} \\
&= -u^* \frac{\partial Y}{\partial x_i} u \\
&= -u^* Y_i u \\
&= -\mathbf{tr}(Y_i u u^*). \qquad \text{(A.4)}
\end{aligned}$$

$$\begin{aligned}
g_{2_i} &= \frac{\partial \phi_2(x)}{\partial x_i} \\
&= u^* \frac{\partial R(W,Y)}{\partial x_i} u \\
&= u^* \frac{\partial}{\partial x_i} \left\{ AY + YA^T + B_2 W + W^T B_2^T + B_1 B_1^T + \right. \\
&\quad \left. (C_1 Y + D_{12} W)(C_1 Y + D_{12} W)^T \right\} u \\
&= u^* \left\{ AY_i + Y_i A^T + B_2 W_i + W_i^T B_2^T + \right. \\
&\quad \left. (C_1 Y_i + D_{12} W_i)(C_1 Y + D_{12} W)^T + (C_1 Y + D_{12} W)(C_1 Y_i + D_{12} W_i)^T \right\} u \\
&= \mathbf{tr} \left(\left\{ AY_i + Y_i A^T + B_2 W_i + W_i^T B_2^T + \right. \right. \\
&\quad \left. \left. (C_1 Y_i + D_{12} W_i)(C_1 Y + D_{12} W)^T + (C_1 Y + D_{12} W)(C_1 Y_i + D_{12} W_i)^T \right\} u u^* \right) \\
&= 2\,\mathbf{tr} \left(\left\{ (C_1 Y + D_{12} W)^T u u^* C_1 + u u^* A \right\} Y_i \right) + \\
&\quad 2\,\mathbf{tr} \left(\left\{ (C_1 Y + D_{12} W)^T u u^* D_{12} + u u^* B_2 \right\} W_i \right). \qquad \text{(A.5)}
\end{aligned}$$

The arrangement of these expressions is not unique, but has been chosen deliberately to place the basis matrices W_i and Y_i on the outside of the argument of the trace. The nature of the basis matrices W_i and Y_i being all zero except a single one

or pair of one's, meant that neither the evaluation of the trace nor the multiplication by W_i or Y_i was required, but simply the selection of a single appropriate element out of the central expression.

In the case of the objective function, the gradient will first be derived for f_2 (max eigenvalue).

$$\begin{aligned}
g_{3_i} &= \frac{\partial \psi(x)}{\partial x_i} = \frac{\partial f_2(M(W,Y))}{\partial x_i} = \frac{\partial \lambda_{max}(M(W,Y))}{\partial x_i} \\
&= u^* \frac{\partial M(W,Y)}{\partial x_i} u \\
&= u^* \frac{\partial}{\partial x_i} \left\{ (C_0 Y + D_{02} W) Y^{-1} (C_0 Y + D_{02} W)^T \right\} u \\
&= u^* \Big\{ (C_0 Y_i + D_{02} W_i) Y^{-1} (C_0 Y + D_{02} W)^T \\
&\quad + (C_0 Y + D_{02} W) Y^{-1} (C_0 Y_i + D_{02} W_i)^T \\
&\quad - (C_0 Y + D_{02} W) Y^{-1} Y_i Y^{-1} (C_0 Y + D_{02} W)^T + \Big\} u \\
&= \mathbf{tr} \left\{ \left(Y^{-1} (C_0 Y + D_{02} W)^T u^* u C_0 + C_0^T u^* u (C_0 Y + D_{02} W) Y^{-1} \right) Y_i \right\} \\
&\quad - \mathbf{tr} \left\{ Y^{-1} (C_0 Y + D_{02} W)^T u^* u (C_0 Y + D_{02} W) Y^{-1} Y_i \right\} \\
&\quad + \mathbf{tr} \left\{ Y^{-1} (C_0 Y + D_{02} W)^T u^* u D_{02} W_i + W_i^T D_{02}^T u^* u (C_0 Y + D_{02} W) Y^{-1} \right\}.
\end{aligned} \tag{A.6}$$

The gradients of $f_1(M(W,Y))$ and $f_3(M(W,Y))$ are then minor modifications [Ref. 29]. For f_1, replace u^*u with the identity matrix I. For f_3, replace u with the elemental vector $e_i = [0, \ldots, 0, 1, 0, \ldots, 0]^T$, where the unit digit corresponds with the position of the maximum diagonal element.

2. Continuous Time $\mathcal{H}_2$ / $\mathcal{H}_\infty$ State-Feedback Synthesis Codes

This section lists the continuous mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ state-feedback controller design codes. The function `h2infsyn` is the principal script. The function `h2infformc` is an administrative script that maps x into W and Y. Finally, the functions `subgrad1c`, `subgrad2c`, and `costgrad1c` calculate the subgradient vectors for

the two constraint functionals and the cost function. The coding of the subgradients was validated by comparison with gradient vectors which were determined by brute force perturbation methods (i.e. $g_i = \frac{f(x+\epsilon_i\delta)-f(x)}{\delta}$, where $\epsilon_i = [0,\ldots,0,1,0,\ldots,0]^T$).

h2infsyn

```
function [K,Kp,X,E,count,Psi1,Psi2,outcome,time]=...
        h2infsyn(f,p,Dim,Sf,mitr,exit,Xi,Ei,counti,Psi1i,Psi2i)

%  function [K,Kp,X,E,count,Psi1,Psi2,outcome,time]=...
%       h2infsyn(f,p,Dim,Sf,mitr,exit,Xi,Ei,counti,Psi1i,Psi2i)
%
%  Finds state-feedback gains for (sub)optimal mixed h2/hinf control.
%  Solves for appropriate fi gains with which to build output feedback
%  controller if input p is auxiliary plant.
%  Method of evaluating generalized mixed h2/hinf cost function can be selected:
%  f=1(trace), 2(max eigenvalue), or 3(max diag element)
%  System matrix 'p' must be a packed mu-tools system matrix:
%
%              Dim(1) Dim(3)  Dim(2)
%       p  =  | AA    | B1        B2    |
%              | C0    | D01       D02   |
%              | C1    | D11       D12   |   Dim(4)
%
%  the columns and rows associated with the input w and the output z1
%       must have been scaled for gamma=1
%  mitr- max number of iterations
%  exit- termination threshold for mixed cost, expressed as a percentage
%  the other inputs are for restart and the code should be consulted
%
%  Outputs:
%  K- dynamic output feedback controller in packed mu-tools format
%  Kp- central controller from Riccati methods
%  count-  Returns number of iterations for each path
%  Psi1-   Upper bound on mixed cost
%  Psi2-   Lower bound on mixed cost
%  outcome- textual result
%  time-   elapsed CPU time
time=cputime;
outcome='max iterat';

%  unpack system matrices
[AA,B,C,D]=unpck(p);
nstates=Dim(1);      pdist=Dim(3);    routput=Dim(4);    sdof=Dim(5);
tt=row(C);           pq=Dim(2)+pdist;
B1=B(:,1:pdist);                    B2=B(:,pdist+1:pq);
C0=C(1:(tt-routput),:);         C1=C((tt-routput+1):tt,:);
D01=D(1:(tt-routput),1:pdist);
```

```
D11=D((tt-routput+1):tt,1:pdist);
D02=D(1:(tt-routput),pdist+1:pq);
D12=D((tt-routput+1):tt,pdist+1:pq);
B1B1=B1*B1';   %  precompute to save flops


%  Determine feasibility/central controller
ham=[AA, ((B1*B1'/0.998)-B2*((D12'*D12)\B2')); -C1'*C1, -AA'];
[x1,x2,fail]=ric_schr(ham);          Xinf=x2/x1;
if (fail>0)
   disp('Statefeedback Hinf problem infeasible'),
   outcome='INFEASIBLE';
   return
elseif min(eig(Xinf))<=0,
   disp('Hinf problem infeasible'),
   outcome='INFEASIBLE';
   return
end
Kp=-(D12'*D12)\B2'*Xinf;   % central controller


%  Determine specific soln from central controller
a1=AA+B2*Kp;    b1=[B1,10000*sqrt(eps)*eye(nstates)];     c1=C1+D12*Kp;
[x1,x2,fail]=ric_schr([a1',c1'*c1;-b1*b1',-a1]);        Yp=x2/x1;
if (fail>0),
   outcome='INDEFINITE';   return
elseif (min(eig(Yp))<=0),
   outcome='INDEFINITE';   return
else,
   disp('State feedback hinf problem feasible'),
end
Wp=Kp*Yp;

%  Initialize problem or use last value?
count=[0 0 0];      Psi1=inf;   Psi2=0;
X=zeros(sdof,1);    E=1000*eye(sdof);
if nargin>8,
   count=counti;    Psi1=Psi1i;   Psi2=Psi2i;
   X=Xi;                E=Ei;
end

% begin ellipsoidal search routine
for k=1:mitr,
   [W,Y]=h2infformc(X,Wp,Yp,Dim);     % maps X to (W,Y)
   [Vy,Ey]=eig(-Y);
   [Phi1,index1]=max(diag(Ey));

   %  Enforce Y>0
   if Phi1>=0,
      g=subgrad1c(Vy(:,index1),nstates,sdof);
      Eg=E*g;        gAg=sqrt(g'*Eg);         Eg=Eg/gAg;
```

```
      if Phi1>gAg,
         outcome='indefinite';   return
      end
      alpha=Phi1/gAg;            count=count+[1 0 0];

   %  Enforce hinf constraint
   else,
      L1=C1*Y+D12*W;
      a=AA*Y+B2*W;
      [Vinf,Einf]=eig(a+a'+B1B1+(L1'*L1));
      [Phi2,index]=max(diag(Einf));
      if Phi2>=0,
         g=subgrad2c(W,Y,Vinf(:,index),Dim,AA,B2,C1,D12,L1);
         Eg=E*g;          gAg=sqrt(g'*Eg);         Eg=Eg/gAg;
         if Phi2>gAg,
            outcome='infeasible';  return
         end
         alpha=Phi2/gAg;         count=count+[0 1 0];

      %  Given the above constraints satisfied, follow cost gradient
      else,
         [Psi,g]=costgradc(f,W,Y,C0,D02,Sf,Dim);
         Eg=E*g;          gAg=sqrt(g'*Eg);          Eg=Eg/gAg;
         Psi2=max([(Psi-gAg);Psi2]);
         if Psi<Psi1,
            Psi1=Psi;   alpha=0;
         else,
            alpha=(Psi-Psi1)/gAg;
         end
         count=count+[0 0 1];

         %  Exit criteria
         if gAg<exit*Psi1,
            disp('Program converged to solution')
            outcome='converged!';
            break,
         end
      end
   end
   X=X-Eg*((1+sdof*alpha)/(sdof+1));
   E=E-2*Eg*(Eg'*((1+sdof*alpha)/(sdof+1)/(1+alpha)));
   E=E*((1-alpha^2)*sdof^2/(sdof^2-1));
   E=(E+E')/2;
end

time=cputime-time;
if outcome=='converged!',
   K=W/Y;
end

% Following are called non-organic functions:
```

```
%  function [W,Y,K2]=h2infformc(X,Wp,Yp,Dim)
%  function g1=subgrad1c(v1,nstates,sdof)
%  function g2=subgrad2c(W,Y,v2,Dim,AA,B2,C1,D12)
%  function [Psi,g]=costgradc(f,W,Y,C0,D02,Sf,Dim)
%  function out=pck(a,b,c,d) from mu-tools
%  function (a,b,c,d)=unpck(p) from mu-tools

%  end h2infsyn

\{verbatim}
\nm

\nid \underline{h2infformc}

\ft
\begin{verbatim}
function [W,Y]=h2infformc(X,Wp,Yp,Dim)

%  reformats the space X in Rs to the two matrices
%  X is assumed to be comprised of the diagonal rows of Y, starting with the
%  main diagonal, followed by W in column order

nstates=Dim(1);  qcontrol=Dim(2);   pdist=Dim(3);   sdof=Dim(5);

j=nstates;
Y=Yp+diag(X(1:nstates));
for k=1:nstates-1,
   i=nstates-k;
   Y=Y+diag(X(j+1:j+i),k)+diag(X(j+1:j+i),-k);
   j=j+i;
end

W=Wp+reshape(X(j+1:j+nstates*qcontrol),qcontrol,nstates);

% end h2infformc
```

subgrad1c

```
function g1=subgrad1c(v,nstates,sdof)

%  calculates subgradient for first constraint function Y>0

g1=zeros(sdof,1);
k=1;

for i=1:nstates                % counts out diag rows
   for j=1:nstates+1-i         % counts down diag rows
      if i==1,
         g1(k)=-v(j)*v(j);
```

```
      else,
         g1(k)=-2*v(j)*v(j+i-1);
      end
      k=k+1;
   end
end

% end subgrad1c
```

subgrad2c

```
function g2=subgrad2c(W,Y,v2,Dim,AA,B2,C1,D12,L1);

%  Uses analytical expression for subgradient
%  See page 16 of journal
nstates=Dim(1);  qcontrol=Dim(2);   sdof=Dim(5);

a=v2*(v2'*(AA+L1'*C1));
Sy=a+a';
Sw=((B2'+D12'*L1)*v2)*v2';

k=1;
g2=zeros(sdof,1);

for i=1:nstates                % counts out diag rows
   for j=1:nstates+1-i         % counts down diag rows
      if i==1,
         g2(k)=Sy(j,j);
      else,
         g2(k)=Sy(j,j+i-1)+Sy(j+i-1,j);
      end
      k=k+1;
   end
end

g2(k:(k+qcontrol*nstates-1))=2*reshape(Sw,qcontrol*nstates,1);

%  end subgrad2C
```

costgradc

```
function [Psi,g3]=costgradc(f,W,Y,C0,D02,Sf,Dim)

%  computes the cost function/gradient for the mixed H2/Hinf continuous time
%  input argument f selects which of the 3 generalized costs is to be
%  implemented:
%          1= trace
%          2= max eigenvalue
```

```
%          3= max diagonal entry

nstates=Dim(1);  qcontrol=Dim(2);   sdof=Dim(5);

L0=C0*Y+D02*W;
M=L0*(Y\L0')+Sf;

if f==1,
   Psi=trace(M);
   b=L0/Y;
   RY=C0'*b+b'*C0-b'*b;
   RW=2*D02'*C0;
else,
   if f==2,
      [V3,E3]=eig(M);
      [Psi,index]=max(diag(E3));
      v=V3(:,index);
   elseif f==3,
      E3=diag(M);
      [Psi,index]=max(E3);
      v=zeros(length(E3),1);  v(index)=1;
   end
   a=C0'*v;
   b=(v'*L0)/Y;
   RY=a*b+b'*a'-b'*b;
   RW=2*(D02'*v)*b;
end

k=1;               g3=zeros(sdof,1);

for i=1:nstates                % counts out diag rows
   for j=1:nstates+1-i         % counts down diag rows
      if i==1,
         g3(k)=RY(j,j);
      else,
         g3(k)=2*RY(j,j+i-1);
      end
      k=k+1;
   end
end

g3(k:(k+qcontrol*nstates-1))=reshape(RW,qcontrol*nstates,1);

% end costgradc
```

3. Continuous Mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ Output-feedback Synthesis Codes

This section lists the design code for the continuous time mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ output-feedback controller synthesis problem. The development follows exactly the formulae of [Ref. 11] in constructing a auxiliary plant, solving the $\mathcal{H}_\infty$ filtering equation by Riccati methods. This auxiliary plant is then fed to the state-feedback controller synthesis code above (`h2infsyn`).

h2infopfb

```
function [K,Psi1,count,outcome,time]=h2infopfb(f,p,Dim,mitr,exit)
%  function [K,Psi1,count,outcome,time]=h2infopfb(f,p,Dim,mitr,exit)
%
%  Finds output feedback controller for (sub)optimal mixed h2/hinf control.
%
%  Method of evaluating generalized mixed h2/hinf cost function can be selected:
%  f=1(trace), 2(max eigenvalue), or 3(max diag element)
%  System matrix 'p' must be a packed mu-tools system matrix:
%
%              Dim(1)  Dim(3)  Dim(2)
%       p   =  | AA    |  B1       B2  |
%              | C0    |  D01      D02 |
%              | C1    |  D11      D12 |  Dim(4)
%              | C2    |  D21      D22 |  Dim(5)
%
%  mitr- max number of iterations
%  exit- termination threshold for mixed cost, expressed as a percentage
%
%  Outputs:
%  K- dynamic output feedback controller in packed mu-tools format
%  Psi1-   Upper bound on mixed cost
%  count-  Returns number of iterations for each path
%  outcome- textual result
%  time-   elapsed CPU time
%
%  formulae are from Rotea and Khargonekar, CDC Proceedings 12/91


% unpack system matrices
[AA,B,C,D]=unpck(p);
nstates=Dim(1);      qcontrol=Dim(2);       pdist=Dim(3);
routput=Dim(4);      moutput=Dim(5);
nout0=row(C)-routput-moutput;               pq=Dim(2)+pdist;
tt=row(D)-moutput;
B1=B(:,1:pdist);                      B2=B(:,pdist+1:pq);
C0=C(1:nout0,:);             C1=C((nout0+1):nout0+routput,:);
```

```
          C2=C((nout0+routput+1):row(C),:);
D01=D(1:nout0,1:pdist);
D11=D((tt-routput+1):tt,1:pdist);
D21=D((nout0+routput+1):row(C),1:pdist);
D02=D(1:(tt-routput),pdist+1:pq);
D12=D((tt-routput+1):tt,pdist+1:pq);
D22=D((nout0+routput+1):row(C),pdist+1:pq);

pnom=pck(AA,B,C(nout0+1:row(C),:),D(nout0+1:row(C),:));

% Designing a pure hinf controller
disp('Designing a pure hinf controller to determine feasibility')

[Khinf,g,gfin,Ax,Ay]=hinfsyn(pnom,moutput,qcontrol,0,1,.01);
if isempty(Khinf),
   disp('Hinf problem is infeasible')
   return
elseif gfin<=1,
   disp('Hinf problem is feasible')
   disp('Gamma value above is measure of freedom for H2 optimization')
end

% Solve the filtering equation
C2tilde=sqrtm(D21*D21')\C2;
ham=[AA' (C1'*C1-C2tilde'*C2tilde);-B1*B1' -AA];
[q1,q2,fail,eig_min]=ric_schr(ham);
if fail>0,
   error('filtering equation troubles')
end
Q=q2/q1;    Q=Q+Q'/2;
if min(abs(eig(Q)))<-1e-10,
   error('Q is not positive definite')
end

% build the auxiliary plant
AAa=AA+Q*C1'*C1;
B1a=Q*C2tilde';
B2a=B2+Q*C1'*D12;
D01=zeros(row(D01),moutput);
D11=zeros(routput,moutput);

paux=pck(AAa,[B1a,B2a],[C0;C1],[D01,D02;D11,D12]);
Sf=C0*Q*C0';
Dima=[Dim(1),col(B2a),col(B1a),row(D01),nstates*(nstates+1)/2+nstates*col(B2a)];

% Design a state-feedback mixed controller for the auxiliary plant
[K1p,Kp,X,E,count,Psi1,Psi2,outcome,time]= h2infsyn(f,paux,Dima,Sf,mitr,exit);

%  Build the dynamic controller
K=pck((AAa-B1a*C2tilde+B2a*K1p),B1a,K1p,zeros(qcontrol,moutput));
```

```
% end h2infopfb
```

C. DISCRETE TIME MIXED $\mathcal{H}_2$ / $\mathcal{H}_\infty$ CONTROLLER SYNTHESIS

As discussed in Chapter III and [Ref. 12], the mixed output-feedback control problem is based upon a convex expression of the full-information problem. It can be solved by the solution of a filtering Riccati equation, and the construction of an auxiliary plant for which a full-information controller is designed. This section consequently develops and lists the state-feedback controller synthesis tools which could either be used independently, or called by the output-feedback synthesis function. Furthermore, in the absence of commercial $\mathcal{H}_\infty$ synthesis tools for discrete time, a measurement-feedback code was written, based on Riccati methods. It was helpful both for assessing the feasibility of example problems, and also formed the structural basis for the mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ synthesis routine. This section consequently concludes with listings of both the discrete $\mathcal{H}_\infty$ and $\mathcal{H}_2$ / $\mathcal{H}_\infty$ synthesis codes for measurement feedback controllers.

1. Numerical Solution of the Discrete Time Full-Information Problem

This section outlines the development of the ellipsoidal design codes for the discrete mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ full-information controller synthesis problem. Recall from Theorem 3.5 that the problem can be expressed as a convex optimization problem:

Minimize:

$$f_i(M(W,Y,K_2)) :=$$

$$f_i\left\{(C_0Y + D_{02}W)Y^{-1}(C_0Y + D_{02}W)^T + (D_{01} + D_{02}K_2)(D_{01} + D_{02}K_2)^T\right\}, \tag{A.7}$$

Subject to: $Y > 0$, and $L(W, Y, K2) < 0$,

where

$$L(W,Y,K2) := \begin{bmatrix} AY + B_2W \\ C_1Y + D_{12}W \end{bmatrix} Y^{-1} \begin{bmatrix} AY + B_2W \\ C_1Y + D_{12}W \end{bmatrix}^T + \begin{bmatrix} B_1 + B_2K_2 \\ D_{11} + D_{12}K_2 \end{bmatrix} \begin{bmatrix} B_1 + B_2K_2 \\ D_{11} + D_{12}K_2 \end{bmatrix}^T - \begin{bmatrix} Y & 0 \\ 0 & I \end{bmatrix}, \quad \text{(A.8)}$$

and f_1 is the trace, f_2 is the maximum eigenvalue, or f_3 is the maximum diagonal element.

The structure of the code is very similar to the continuous codes above. Two items are of significance which warrant independent discussion. First of all, the structure of the controller is different for the discrete problem, in that it is now a full-information feedback gain matrix, rather than state-feedback. The parameterization of the controller is now $K_{fi} = [WY^{-1} \quad K2]$, necessitating that the search has an additional $p \times q$ degrees of freedom, where $p := \dim(w)$ and $q := \dim(u)$. A set of basis matrices $K_{2_i} \in \mathbf{R}^{q \times p}$ was consequently constructed with identical structure as the set of W_i above. The second, and more challenging issue was the determination of an initial feasible controller. This was necessary in order to assess the feasibility of the problem and to give the iterative ellipsoidal search a good starting point. To this date, no commercial codes are available to solve the discrete time $\mathcal{H}_\infty$ Riccati equations, and so considerable effort was devoted to cleanly solving these problems. The codes for these problems are included here as subroutines to the main script.

1. Determine if the $\mathcal{H}_\infty$ problem is feasible, and determine a particular feasible controller $K_p = [K_{1_p} \quad K_{2_p}]$. This was done by solving the discrete time $\mathcal{H}_\infty$ synthesis equation [Ref. 31].

2. Determine Y_p and $W_p = K_p Y_p$ from the $\mathcal{H}_\infty$ analysis equation (3.2).

3. Initialize the search at $x = [0, \ldots, 0]^T$, with a very large ellipsoid about x.

4. From$x$, determine Y, W, and K_2 as perturbations of the particular solution (W_p, Y_p, K_{2_p}).

5. Evaluate the constraint $\phi_1(x) = \lambda_{max}(-Y) < 0$. If $\phi_1(x) \geq 0$, then use the eigenvector associated with the maximum eigenvalue to calculate the subgradient: $g_1 = \frac{\partial \phi_1(x)}{\partial x}$. Update the ellipsoid and x and return to step 4.

6. Evaluate the constraint $\phi_2(x) = \lambda_{max}(L(W, Y, K2)) < 0$. If $\phi_2(x) \geq 0$, then use the eigenvector associated with the maximum eigenvalue to calculate the subgradient: $g_2 = \frac{\partial \phi_2(x)}{\partial x}$. Update the ellipsoid and x and return to step 4.

7. With both constraint functionals satisfied, evaluate the objective function $\psi(x) = f(W, Y, K_2))$ and its subgradient: $g_3 = \frac{\partial \psi(x)}{\partial x}$. Update the ellipsoid, x, and the upper and lower bounds on the estimate of the optimum cost.

8. Return to step 4 unless the termination criteria is satisfied.

9. Calculate the (sub)optimal controller $K = [WY^{-1} \quad K_2]$.

10. Exit.

The discussion above in section B, regarding computational issues, is also germane to these codes.

The analytical expressions for the subgradient expressions are derived below. Recall that u is the eigenvector associated with the maximum eigenvalue. The first subgradient g_1 is omitted since it is identical to the continuous time problem. Due to the structure of $L(W, Y, K2)$, it is easiest to break the gradient expression for

g_2 into its respective parts. Additionally, the replacement $K_1 = WY^{-1}$ is made when convenient.

For $i = 1$ to $\frac{n(n+1)}{2} + nq$, $K_{2_i} = 0$:

$$
\begin{aligned}
g_{2_i} &= \frac{\partial \phi_2(x)}{\partial x_i} \\
&= u^* \frac{\partial L(W, Y, K2)}{\partial x_i} u \\
&= u^* \frac{\partial}{\partial x_i} \left\{ \begin{bmatrix} AY + B_2 W \\ C_1 Y + D_{12} W \end{bmatrix} Y^{-1} \begin{bmatrix} AY + B_2 W \\ C_1 Y + D_{12} W \end{bmatrix}^T + - \begin{bmatrix} Y & 0 \\ 0 & I \end{bmatrix} \right\} u \\
&= u^* \left\{ \begin{bmatrix} A & B_2 \\ C_1 & D_{12} \end{bmatrix} \begin{bmatrix} Y_i \\ W_i \end{bmatrix} Y^{-1} \begin{bmatrix} AY + B_2 W \\ C_1 Y + D_{12} W \end{bmatrix}^T \right. \\
&\quad + \begin{bmatrix} AY + B_2 W \\ C_1 Y + D_{12} W \end{bmatrix} Y^{-1} \begin{bmatrix} Y_i \\ W_i \end{bmatrix}^T \begin{bmatrix} A & B_2 \\ C_1 & D_{12} \end{bmatrix}^T \\
&\quad \left. - \begin{bmatrix} AY + B_2 W \\ C_1 Y + D_{12} W \end{bmatrix} Y^{-1} Y_i Y^{-1} \begin{bmatrix} AY + B_2 W \\ C_1 Y + D_{12} W \end{bmatrix}^T - \begin{bmatrix} Y_i & 0 \\ 0 & 0 \end{bmatrix} \right\} u \\
&= 2\,\mathbf{tr} \left\{ \begin{bmatrix} A + B_2 K_1 \\ C_1 + D_{12} K_1 \end{bmatrix}^T uu^* \begin{bmatrix} A & B_2 \\ C_1 & D_{12} \end{bmatrix} \begin{bmatrix} Y_i \\ W_i \end{bmatrix} \right\} \\
&\quad - \mathbf{tr} \left\{ \begin{bmatrix} A + B_2 K_1 \\ C_1 + D_{12} K_1 \end{bmatrix}^T uu^* \begin{bmatrix} A + B_2 K_1 \\ C_1 + D_{12} K_1 \end{bmatrix}^T Y_i - uu^* \begin{bmatrix} Y_i & 0 \\ 0 & 0 \end{bmatrix} \right\}. \qquad \text{(A.9)}
\end{aligned}
$$

For $i = \frac{n(n+1)}{2} + nq + 1$ to $\frac{n(n+1)}{2} + (n+p)q$, $Y_i = 0$ and $W_i = 0$:

$$
\begin{aligned}
g_{2_i} &= \frac{\partial \phi_2(x)}{\partial x_i} \\
&= u^* \frac{\partial L(W, Y, K2)}{\partial x_i} u \\
&= u^* \frac{\partial}{\partial x_i} \left\{ \begin{bmatrix} B_1 + B_2 K_2 \\ D_{11} + D_{12} K_2 \end{bmatrix} \begin{bmatrix} B_1 + B_2 K_2 \\ D_{11} + D_{12} K_2 \end{bmatrix}^T \right\} u \\
&= u^* \left\{ \begin{bmatrix} B_2 \\ D_{12} \end{bmatrix} K_{2_i} \begin{bmatrix} B_1 + B_2 K_2 \\ D_{11} + D_{12} K_2 \end{bmatrix}^T + \begin{bmatrix} B_1 + B_2 K_2 \\ D_{11} + D_{12} K_2 \end{bmatrix} K_{2_i}^T \begin{bmatrix} B_2 \\ D_{12} \end{bmatrix}^T \right\} u \\
&= 2\mathbf{tr} \left\{ \begin{bmatrix} B_1 + B_2 K_2 \\ D_{11} + D_{12} K_2 \end{bmatrix}^T uu^* \begin{bmatrix} B_2 \\ D_{12} \end{bmatrix} K_{2_i} \right\}. \qquad \text{(A.10)}
\end{aligned}
$$

The subgradient for the cost functional will first be derived for f_2, and then

modified as above:

$$
\begin{aligned}
g_{3_i} &= \frac{\partial \phi_2(x)}{\partial x_i} \\
&= u^* \frac{\partial M(W,Y,K2)}{\partial x_i} u \\
&= u^* \frac{\partial}{\partial x_i} \left\{ (C_0 Y + D_{02} W) Y^{-1} (C_0 Y + D_{02} W)^T \right. \\
&\quad \left. + (D_{01} + D_{02} K_2)(D_{01} + D_{02} K_2)^T \right\} u \\
&= u^* \left\{ [C_0 \;\; D_{02}] \begin{bmatrix} Y_i \\ W_i \end{bmatrix} (C_0 + D_{02} K_1)^T + (C_0 + D_{02} K_1) [Y_i \;\; W_i^T] \begin{bmatrix} C_0^T \\ D_{02}^T \end{bmatrix} \right\} u \\
&\quad - u^* \left\{ (C_0 + D_{02} K_1) Y_i (C_0 + D_{02} K_1)^T \right\} u \\
&\quad u^* \left\{ + D_{02} K_{2_i} (D_{01} + D_{02} K_2)^T + (D_{01} + D_{02} K_2) K_{2_i}^T D_{02}^T \right\} u \\
&= 2 \, \mathbf{tr} \left\{ (C_0 + D_{02} K_1)^T u u^* [C_0 \;\; D_{02}] \begin{bmatrix} Y_i \\ W_i \end{bmatrix} \right\} \\
&\quad + \mathbf{tr} \left\{ (C_0 + D_{02} K_1)^T u u^* (C_0 + D_{02} K_1) Y_i \right\} \\
&\quad + 2 \, \mathbf{tr} \left\{ (D_{01} + D_{02} K_2)^T u u^* D_{02} K_{2_i} \right\}. \qquad \text{(A.11)}
\end{aligned}
$$

The gradients of $f_1(W,Y,K_2))$ and $f_3(W,Y,K_2)$ are then minor modifications [Ref. 29]. For f_1, replace u^*u with the identity matrix I. For f_3, replace u with the elemental vector $\epsilon_i = [0,\ldots,0,1,0,\ldots,0]^T$, where the unit digit corresponds with the position of the maximum diagonal element.

2. Discrete Time $\mathcal{H}_2$ / $\mathcal{H}_\infty$ Full-Information Controller Synthesis Codes

This section lists the discrete mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ state-feedback controller design codes. The function `dh2infsyn1` is the principal script. The function `h2infform` is an administrative script that maps the vector x into W, Y, and K_2. The functions `dfiric2` and `dhinfric` solve the discrete $\mathcal{H}_\infty$ synthesis and analysis Riccati equations. Finally, the functions `subgrad1`, `subgrad2`, and `costgrad1c` calculate the subgradient vectors for the two constraint functionals and the cost function. Note that `subgrad1` is identical to its continuous time counterpart and is included

here simply for completeness. The coding of the subgradients was the difficult part of the development, and the codes were validated (and the multiple errors corrected) by comparison with gradient vectors which were determined by brute force perturbation methods (i.e., $g_i = \frac{f(x+e_i\delta)-f(x)}{\delta}$, where $e_i = [0,\ldots,0,1,0,\ldots,0]^T$).

dh2infsyn1

```
 function [K,Kp,X,E,count,Psi1,Psi2,outcome,time]=...
       dh2infsyn1(f,pd,Dim,Sf,mitr,exit,Xi,Ei,counti,Psi1i,Psi2i)

% function [K,Kp,X,E,count,Psi1,Psi12,outcome,time]=...
%      dh2infsyn1(f,pd,Dim,Sf,mitr,exit,Xi,Ei,counti,Psi1i,Psi2i)
%
%  Finds full information feedback gains for DISCRETE (sub)optimal mixed h2/hinf
%  control. Solves for appropriate fi gains with which to build output feedback
%  controller if input 'pd' is auxiliary plant.
%  Hinf constraint is applied as infnorm(Tz1w)<1.  Different values of gamma
%  must be absorbed into the appropriate rows of the plant matrix.
%
%  Inputs:
%  Method of evaluating generalized mixed h2/hinf cost function can be selected:
%  f=1(trace), 2(max eigenvalue), or 3(max diag element)
%  System matrix 'pd' must be a packed mu-tools system matrix,with the noted
%  dimensions:
%
%              Dim(1)   Dim(3)  Dim(2)
%              | AA     |  B1      B2  |
%       pd=    | C0     |  D01     D02 |
%              | C1     |  D11     D12 |  Dim(4)
%
%  Note: the columns and rows associated with the input w and the output z1
%        must have been scaled for gamma=1
%  'Dim' holds the descriptions of the problem size:
%     Dim(5)=sdof=Dim(1)*(Dim(1)+1)/2 + Dim(2)*(Dim(1)+Dim(2))
%  'Sf' is the filtering cost matrix for the output feedback problem and
%     should be set to zeros(nstates) for full info feedback problem
%  'mitr' is the max number of iterations
%  'exit' is the exit criteria as fraction of the cost
%  The remaining input variables are for restarting a problem that had not yet
%     reached convergence and should be empty for initialization.
%
%  The outputs are the gain matrix K=[K1,K2], the central controller Kp, and the
%  stopping parametrics which provide for restart capability if the problem had
%  not yet converged. 'outcome' identifies the termination branch followed.
%
%  Constraints are applied using the formulation of Thm 4.3 from Kaminer,
%  Khargonekar and Rotea.  This is the TWO STEP contraint.
%
```

```
%  This version uses MATLAB's 'eig' to evaluate positive definiteness

%time=cputime;
outcome='max iterat';

%  unpack system matrices
[AA,B,C,D]=unpck(pd);
sdof=Dim(5);     nstates=Dim(1);
pdist=Dim(3);    routput=Dim(4);    tt=row(C);     pq=Dim(2)+pdist;
B1=B(:,1:pdist);                    B2=B(:,pdist+1:pq);
C0=C(1:(tt-routput),:);             C1=C((tt-routput+1):tt,:);
D01=D(1:(tt-routput),1:pdist);
D11=D((tt-routput+1):tt,1:pdist);
D02=D(1:(tt-routput),pdist+1:pq);
D12=D((tt-routput+1):tt,pdist+1:pq);

%  Determine feasibility and specific solution (central controller)
%  Solve the synthesis Riccati equation
[K1p,K2p,P,Perr,erflg]=dfiric2(AA,B1,B2,C1,D11,D12,0.99);
if erflg>1,
   outcome='INFEASIBL1'; return
end
Kp=[K1p,K2p];
% Solve the analysis equation
[Yp,erflg]=dhinfric((AA+B2*K1p)',(C1+D12*K1p)',(B1+B2*K2p)',(D11+D12*K2p)',0.99);
if erflg>0,
   outcome='INFEASIBL2'; return
end
Wp=K1p*Yp;

%  Initialize problem or use last value?
%  This is a weak point in the code right now as it initializes the ellipsoid
%  as something arbitrarily huge.  Need to find a way to initialize the ellipsoid
%  in a smarter way.

count=[0 0 0];        Psi1=inf;         Psi2=0;  thresh=0;
X=zeros(sdof,1);      E=100*eye(sdof);
if nargin>6,
   count=counti;      Psi1=Psi1i;     Psi2=Psi2i;
   X=Xi;              E=Ei;
elseif nargin<6,
   error('insufficient number of input arguements')
end
Z=zeros(nstates,routput);              gam=1;

for k=1:mitr,
   [W,Y,K2]=h2infform(X,Wp,Yp,K2p,Dim);     % maps X to (W,Y,K2)

   %  Enforce Y>0
   [Vy,Ey]=eig(-Y);
   [Phi1,index1]=max(diag(Ey));
```

```
   if Phi1>0,
      g=subgrad1c(Vy(:,index1),nstates,sdof);
      Eg=E*g;       gAg=sqrt(g'*Eg);       Eg=Eg/gAg;
      if Phi1>gAg,
         outcome='indefinite';  return
      end
      alpha=Phi1/gAg;
      count=count+[1 0 0];

   %  Enforce infinity norm constraint
   else,
      K1=W/Y;
      a=[AA+B2*K1;C1+D12*K1];
      b=[B1+B2*K2;D11+D12*K2];
      L=a*Y*a'+b*b'-[Y,Z;Z',eye(routput)];
      [Vinf,Einf]=eig(L);      [Phi2,index]=max(diag(Einf));
      if Phi2>thresh,
         g=subgrad2(K1,K2,Vinf(:,index),Dim,AA,B2,C1,D12,a,b);
         Eg=E*g;           gAg=sqrt(g'*Eg);          Eg=Eg/gAg;
         if Phi2>gAg,
            outcome='infeasible';  return
         end
         alpha=Phi2/gAg;
         count=count+[0 1 0];

      %  Given above are satisfied, follow cost gradient
      else,
         [Psi,g]=costgrad(f,Y,K1,K2,C0,D01,D02,Dim,Sf);
         Eg=E*g;           gAg=sqrt(g'*Eg);          Eg=Eg/gAg;
         Psi2=max([(Psi-gAg);Psi2]);
         if Psi<Psi1,
            Psi1=Psi;   alpha=0;
         else
            alpha=(Psi-Psi1)/gAg;
         end
         count=count+[0 0 1];

         %  Exit criteria
         if gAg<exit*Psi1,
            disp('Program converged to solution')
            outcome = 'converged!';   break
         end
      end
   end
   X=X-Eg*((1+sdof*alpha)/(sdof+1));
   E=E-2*Eg*(Eg'*((1+sdof*alpha)/(sdof+1)/(1+alpha)));
   E=E*((1-alpha^2)*sdof^2/(sdof^2-1));
   E=(E+E')/2;   % enforce symmetry
end

%time=cputime-time;
```

```
if outcome == 'converged!',
   K=[K1 K2];
end

% Following are called non-organic functions:
%  function [W,Y,K2]=h2infform(X,Wp,Yp,K2p,Dim)
%  function g1=subgrad1(v1,nstates,sdof)
%  function g=subgrad2(K1,K2,Vinf(:,index),Dim,AA,B2,C1,D12,a,b);
%  function [Psi,g]=costgrad(f,Y,K1,K2,C0,D01,D02,Dim,Sf);
%  function Sys=pck(a,b,c,d)   from mu-tools toolbox
%  function [F1,F2,P,Perror,erflg]=dfiric2(A,B1,B2,C,D11,D12,gam);
%  function [Yp,erflg]=dhinfric(F,G,H,J,gam);

%  end dh2infsyn1
```

dfiric2

```
function [F1,F2,P,Perror,erflg]=dfiric2(A,B1,B2,C,D11,D12,gam);

%        [F1,F2,P,Perr,erflg] = DFIRIC2(A,B1,B2,C,D11,D12,gam);
%
% This routine solves the Discrete Algebraic Riccati equation
% for the full information h-infinity problem:
% Find a P such that:
%   V(P) = B2'PB2+D11'D11 > 0
%   R(P)=gam^2*I-D11'D11-B1'PB1+(B1'PB2+D11'D12)*inv(V(P))*(B1'PB2+D11'D12)'>0
%
%       and the DARE:    P = A'PA + C'C - Xb*inv(G(P))*Xb'
%
%       is satisfied with:
%       G(P) = [ D11'D11 + B1'PB1- gam^2I   D11'D12 + B1'PB2  ;
%                D12'D11 + B2'PB1           D12'D12 + B2'PB2 ]
%       Xb  =  A'P[B1 B2]+C'[D11 D12];
%
% where u(k) = [F1,F2]*[x(k)' w(k)']' the solution to the full information
% case. The solution is based on Iglesias' symplectic pencil formulation.
% WARNING: The input argument order is different from Stoorvogel's'dfiric' to
% make the formulation consistent with Rotea and Kaminer's notation.
%
% The following inorganic matlab functions are called:
%   abcdchk2

% begin dfiric2

 erflg=0;

%         CHECK THE CONSISTENCY OF THE MODEL MATCHING PROBLEM

 [m1,n,p,msg1]=abcdchk2(A,B1,C,D11);
```

```
 [m2,n,p,msg2]=abcdchk2(A,B2,C,D12);

 if ~(isempty(msg1) & isempty(msg2)),
   disp('ERROR IN THE REALIZATION');
   disp(msg1),disp(msg2)
   erflg=2;
   return;
 end

Ip=eye(m1);      Ir=eye(p);         In=eye(n);
Zn=zeros(n);
R1=inv([D11'*D11-Ip*gam^2, D11'*D12; D12'*D11, D12'*D12]);

%  symplectic pencil for DARE:
SINF1=[  A-[B1 B2]*R1*[D11 D12]'*C,          Zn
       -C'*(Ir-[D11 D12]*R1*[D11 D12]')*C,   In ];
SINF2=[  In ,    [B1 B2]*R1*[B1 B2]'
         Zn ,    (A-[B1 B2]*R1*[D11 D12]'*C)'];

[Vs,Ts]=eig(SINF1,SINF2);

Ts=diag(Ts);

if min(abs(log(abs(Ts))))<(10000*eps),
   disp('Hamiltonian is not in dom(Ric)'),
   erflg=3;
   return
else,
   index1=find(abs(Ts)<1);
   P=Vs(n+1:2*n,index1)/Vs(1:n,index1);
   P=real(P);
   P=(P+P')/2;                        % Ensures P=P'
end

%  Verify accuracy of solution
GP=inv(inv(R1)+[B1 B2]'*P*[B1 B2]);
Xb=A'*P*[B1 B2]+C'*[D11 D12];
Perror= A'*P*A - P + C'*C - Xb*GP*Xb';

accuracy=(norm(Perror)/norm(P));
if accuracy>1e-6,
   disp(' Warning: Solution to DARE may be inaccurate');
   accuracy
   erflg=1;
end

%  Check for stability
Acloseloop=A-[B1 B2]*GP*Xb';
if max(abs(eig(Acloseloop)))>=1,
   disp('Closed loop system not stable')
   erflg=4;
```

```
   return
end

VPi = pinv(B2'*P*B2+D12'*D12);
F1 = -VPi*(B2'*P*A+D12'*C);
F2 = -VPi*(B2'*P*B1+D12'*D11);
return

%  end dfiric2
```

dhinfric

```
function [Y,erflg]=dhinfric(F,G,H,J,gam);

%        [Y,erflg]=dhinfric(F,G,H,J,gam);
%
% This routine solves the Hinf Discrete Algebraic Riccati equation for a closed
% loop system  F,G,H,J.
%
% Find a Y such that:
%   R = gam^2*I - J'*J - G'*Y*G > 0
%   FYF' - Y + GG' + (FYH' + GJ')*R\(HYF' + JG') = 0
%
%  The solution is based on Iglesias' symplectic pencil formulation (eqn 2.6)
%  Calls chol2 and row from ktools

[n m]=size(G);
R=gam^2*eye(m)-J'*J;      F=F+G*(R\J')*H;

%  form Symplectic pencil
S1=[F zeros(n,n); -H'*inv(eye(row(J))-J*J')*H eye(n)];
S2=[eye(n) -G*(R\G'); zeros(n,n) F'];
[Vs,Ts]=eig(S1,S2);           Ts=diag(Ts);      index1=find(abs(Ts)<1);

if min(abs(log(abs(Ts))))<(10000*eps),
   erflg=1;   return
end
Y=real(Vs(n+1:2*n,index1)/Vs(1:n,index1));   Y=(Y+Y')/2;
if (chol2(Y,n)~=1) | (chol2((R-G'*Y*G),m)~=1),
   erflg=1;   return
end
erflg=0;

% end dhinfric
```

h2infform

```
function [W,Y,K2]=h2infform(X,Wp,Yp,Khinf2,Dim)

%  reformats the space X in Rs to the three matrices
%  X is assumed to be comprised of the diagonal rows of Y, starting with the
%  main diagonal, followed by W in column order, and lastly by K2 by column

nstates=Dim(1);  qcontrol=Dim(2);   pdist=Dim(3);   sdof=Dim(5);

j=nstates;
Y=Yp+diag(X(1:nstates));
for k=1:nstates-1,
   i=nstates-k;
   Y=Y+diag(X(j+1:j+i),k)+diag(X(j+1:j+i),-k);
   j=j+i;
end

W=Wp+reshape(X(j+1:j+nstates*qcontrol),qcontrol,nstates);
K2=Khinf2+reshape(X(j+nstates*qcontrol+1:sdof),qcontrol,pdist);

% end h2infform
```

subgrad1

```
function g1=subgrad1(v,nstates,sdof)

%  calculates subgradient for first constraint function Y>0

g1=zeros(sdof,1);
k=1;

for i=1:nstates                % counts out diag rows
   for j=1:nstates+1-i         % counts down diag rows
      if i==1,
         g1(k)=-v(j)*v(j);
      else,
         g1(k)=-2*v(j)*v(j+i-1);
      end
      k=k+1;
   end
end

% end subgrad1
```

subgrad2

```
function g2=subgrad2(K1,K2,v2,Dim,AA,B2,C1,D12,a,b);
```

```
%  Uses analytical expression for subgradient
%  See page 16 of journal
nstates=Dim(1);  qcontrol=Dim(2);   pdist=Dim(3);   sdof=Dim(5);

ab=a'*v2;
ac=ab*(v2'*[AA;C1]);
bd=(v2'*[B2;D12]);
Sy=ac+ac'-ab*ab';
Sw=ab*bd;
Sk=(b'*v2)*bd;

k=1;
g2=zeros(sdof,1);

for i=1:nstates                % counts out diag rows
   for j=1:nstates+1-i         % counts down diag rows
      if i==1,
         g2(k)=Sy(j,j)-v2(k)^2;
      else,
         g2(k)=Sy(j,j+i-1)+Sy(j+i-1,j)-2*v2(j)*v2(j+i-1);
      end
      k=k+1;
   end
end

g2(k:(k+qcontrol*nstates-1))=2*reshape(Sw,qcontrol*nstates,1);
g2((k+qcontrol*nstates):sdof)=2*reshape(Sk,qcontrol*pdist,1);

%  end subgrad2
```

costgrad

```
function [Psi3,g3]=costgrad(f,Y,K1,K2,C0,D01,D02,Dim,Sf)

%  function [Psi3,g3]=costgrad(f,Y,K1,K2,C0,D01,D02,Dim,Sf)
%
%  FOR USE WITH DH2INFSYN
%  computes the cost function for the mixed H2/Hinf discrete time
%  input argument f selects which of the 3 generalized costs is to be
%  implemented:
%          1= trace
%          2= max eigenvalue
%          3= max diagonal entry
%  also determines the subgradient for the cost
%
%  calls row (ktools)

sdof=Dim(5);    nstates=Dim(1);   qcontrol=Dim(2);    pdist=Dim(3);
```

```
dk1=D02*K1;
a=C0+dk1;
b=D01+D02*K2;
M=a*Y*a'+b*b'+Sf;
if f==1,
   Psi3=trace(M);
   RY=C0'*C0-dk1'*dk1;
   RW=2*D02'*a;
   RK2=2*D02'*b;
else,
   if f==2,
      [V3,E3]=eig(M);
      [Psi3,index]=max(diag(E3));      v=V3(:,index);
   elseif f==3,
      [Psi3,index]=max(diag(M));       v=zeros(row(M),1);  v(index)=1;
   else,
      error('first input argument must be 1,2,or 3')
   end
   a=C0'*v;
   d=D02'*v;
   dk1=d'*K1;
   RK2=2*d*(v'*D01+d'*K2);
   RY=a*a'-dk1'*dk1;
   RW=2*d*(a'+dk1);
end

k=1;              g3=zeros(sdof,1);

for i=1:nstates                % counts out diag rows
   for j=1:nstates+1-i         % counts down diag rows
      if i==1,
         g3(k)=RY(j,j);
      else,
         g3(k)=2*RY(j,j+i-1);
      end
      k=k+1;
   end
end

g3(k:(k+qcontrol*nstates-1))=reshape(RW,qcontrol*nstates,1);
g3((k+qcontrol*nstates):sdof)=reshape(RK2,qcontrol*pdist,1);

% end costgrad
```

3. Discrete $\mathcal{H}_\infty$ Output-Feedback Controller Synthesis Problem

The following design code was prepared to solve the discrete $\mathcal{H}_\infty$ output-feedback controller synthesis problem. In the absence of commercial codes to solve this problem, this code was useful to assess the feasibility of synthesis models prior to attempting the mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ problem. Specifically, the value of achievable γ for the $\mathcal{H}_\infty$ controller was a measure of the degree of freedom available to the $\mathcal{H}_2$ optimization. The $\mathcal{H}_\infty$ problem is solved by solution of the filtering equation by Iglesias' symplectic pencil method [Ref. 47], and then construction of an auxiliary plant per Stoorvogel [Ref. 31].

dhinfsyn

```
function [k,gfin]=dhinfsyn(pd,qcontrol,moutput,logam,higam,tol)

%  function [k,gfin]=dhinfsyn(pd,qcontrol,moutput,logam,higam,tol)
%
%  Determines discrete time controller which minimizes the infinity
%  norm of w to z (to within the specified tolerance).
%  The discrete system P is partitioned (per mu-tools):
%                          | a   b1   b2    |
%              p    =      | c1  d11  d12   |
%                          | c2  d21  d22   |
%   where b2 has column size of the number of control inputs (qcontrol)
%   and c2 has row size of the number of measurements (moutput) being
%   provided to the controller.

%   dhinfsyn calls dfiric2, row, col (personal codes) and unpck from mu-tools
%   toolbox

[AA,BB,CC,DD]=unpck(pd);
nz1=row(CC)-moutput;              nw1=col(BB)-qcontrol;
B1=BB(:,1:nw1);                   B2=BB(:,nw1+1:col(BB));
C1=CC(1:nz1,:);                   C2=CC(nz1+1:row(CC),:);
D11=DD(1:nz1,1:nw1);              D12=DD(1:nz1,nw1+1:col(BB));
D21=DD(1+nz1:row(CC),1:nw1);      D22=DD(1+nz1:row(CC),nw1+1:col(BB));

gam1=higam;     gam2=logam;       gam=gam1;
Ia=eye(D11*D11')-D11*D11';

while (gam1-gam2)/gam1>tol,
```

```
   %  solve the filtering equation
   [L1,L2,Q,Qerror,erflg1]=dfiric2(AA',C1',C2',B1',D11',D21',gam);
   if erflg1<=1,
      %  Build Gfi(Q), the augmented plant
      V=C2*Q*C2'+D21*D21';
      R=Ia-C1*Q*C1'+C1*Q*C2'*(V\C2*Q*C1');
      Z=AA*Q*C1'+B1*D11'-(AA*Q*C2'+B1*D21')*(V\(C2*Q*C1'+D21*D11'));
      Vhalf=inv(sqrtm(V));
      Rhalf=inv(sqrtm(R));

      AAq=AA+Z*(R\C1);
      B1q=(AA*Q*C2'+B1*D21'+Z*(R\(C1*Q*C2'+D11*D21')))*Vhalf;
      B2q=B2+Z*(R\D12);
      C1q=Rhalf*C1;
      D11q=Rhalf*(C1*Q*C2'+D11*D21')*Vhalf;
      D12q=Rhalf*D12;

      %  Now solve for the full-info gains which satisfy the hinf prob
      [K1a,K2a,P,Perror,erflg2]=dfiric2(AAq,B1q,B2q,C1q,D11q,D12q,gam);
      if erflg2<=1,
         gam1=gam;
         gam=(gam+gam2)/2;   disp(gam)
      end
   end

   if max([erflg1;erflg2])>=1,
      if gam==higam,
         error('problem not feasible within specified gamma range')
      end
      gam2=gam;
      gam=(gam+gam1)/2;   disp(gam)
   else,
      K1=K1a;    K2=K2a;
   end
end

%  Build hinf controller
Ac=AAq+B2q*K1-(B1q+B2q*K2)*Vhalf*C2;
Bc=(B1q+B2q*K2)*Vhalf;
Cc=K1-K2*Vhalf*C2;
Dc=K2*Vhalf;
k=pck(Ac,Bc,Cc,Dc);
gfin=gam1;

%  end dhinfsyn
```

4. Discrete $\mathcal{H}_2$ / $\mathcal{H}_\infty$ Output-Feedback Controller

The following function file solves the discrete $\mathcal{H}_2$ / $\mathcal{H}_\infty$ output-feedback controller synthesis problem. It is based upon the optimal discrete $\mathcal{H}_\infty$ code above and differs principally in that after the auxiliary plant is constructed it calls the dh2infsyn to solve the mixed full-information state-feedback controller problem. As with the continuous codes, w and z are assumed to have been scaled in order that the constraint $\| T_{z_1 w} \|_\infty < 1$ is feasible.

dh2infsyn

```
function [K,count,Psi1,Psi2,outcome,time]=dh2infopfb(f,pd,Dim,mitr,exit)

%  Finds full information feedback gains for DISCRETE (sub)optimal mixed h2/hinf
%  control. Solves for appropriate fi gains with which to build output feedback
%  controller if input 'pd' is auxiliary plant.
%  Hinf constraint is applied as infnorm(Tz1w)<1.  Different values of gamma
%  must be absorbed into the appropriate rows of the plant matrix.
%
%  Inputs:
%  Method of evaluating generalized mixed h2/hinf cost function can be selected:
%  f=1(trace), 2(max eigenvalue), or 3(max diag element)
%  System matrix 'pd' must be a packed mu-tools system matrix,with the noted
%  dimensions:
%
%              Dim(1)  Dim(3)  Dim(2)
%       p   =  | AA    |  B1       B2  |
%              | C0    |  D01      D02 |
%              | C1    |  D11      D12 |  Dim(4)
%              | C2    |  D21      D22 |  Dim(5)
%
%  Note: the columns and rows associated with the input w and the output z1
%        must have been scaled for gamma=1
%  'mitr' is the max number of iterations
%  'exit' is the exit criteria as fraction of the cost
%
%   Outputs:
%   K- dynamic controller in packed (mutools) form
%   count- iteration breakdown on ellipsoidal routine
%   Psi1,Psi2- upper/lower bounds for mixed cost

%   dh2infopfb calls dfiric2, row, col (personal codes) and unpck from mu-tools
%   toolbox as well as d2hinfsyn and its subroutines
```

```
% unpack system matrices
[AA,B,C,D]=unpck(p);
nstates=Dim(1);      qcontrol=Dim(2);      pdist=Dim(3);
routput=Dim(4);      moutput=Dim(5);
nout0=row(C)-routput-moutput;              pq=Dim(2)+pdist;
tt=row(D)-moutput;
B1=B(:,1:pdist);                    B2=B(:,pdist+1:pq);
C0=C(1:nout0,:);            C1=C((nout0+1):nout0+routput,:);
          C2=C((nout0+routput+1):row(C),:);
D01=D(1:nout0,1:pdist);
D11=D((tt-routput+1):tt,1:pdist);
D21=D((nout0+routput+1):row(C),1:pdist);
D02=D(1:(tt-routput),pdist+1:pq);
D12=D((tt-routput+1):tt,pdist+1:pq);
D22=D((nout0+routput+1):row(C),pdist+1:pq);


gam=1;
Ia=eye(D11*D11')-D11*D11';

   %  solve the filtering equation
   [L1,L2,Q,Qerror,erflg1]=dfiric2(AA',C1',C2',B1',D11',D21',gam);
   if erflg1<=1,

      %  Build Gfi(Q), the augmented plant
      V=C2*Q*C2'+D21*D21';
      R=Ia-C1*Q*C1'+C1*Q*C2'*(V\C2*Q*C1');
      Z=AA*Q*C1'+B1*D11'-(AA*Q*C2'+B1*D21')*(V\(C2*Q*C1'+D21*D11'));
      Vhalf=inv(sqrtm(V));
      Rhalf=inv(sqrtm(R));

      AAq=AA+Z*(R\C1);
      B1q=(AA*Q*C2'+B1*D21'+Z*(R\(C1*Q*C2'+D11*D21')))*Vhalf;
      B2q=B2+Z*(R\D12);
      D01q=(D01*D21'+C0*Q*C2')*Vhalf;
      C1q=Rhalf*C1;
      D11q=Rhalf*(C1*Q*C2'+D11*D21')*Vhalf;
      D12q=Rhalf*D12;

      pq=pck(AAq,[B1q,B2q],[C0;C1q],[D01q,D02;D11q,D12q]);

     % compute the filtering cost
     Sf=C0*Q*C0+D01*D01'-D01q*D01q;

      %  Now solve for the full-info gains which satisfy the hinf prob
      [Kfi,Kp,X,E,count,Psi1,Psi2,outcome,time] =dh2infsyn1(f,pq,Dim,Sf,mitr,exit);

      K1=Kfi(:,1:nstates);    K2=Kfi(:,nstates+1:nstates+pdist);
   else,
      error('Filtering equation infeasible')
```

```
    return
end


%  Build controller
Ac=AAq+B2q*K1-(B1q+B2q*K2)*Vhalf*C2;
Bc=(B1q+B2q*K2)*Vhalf;
Cc=K1-K2*Vhalf*C2;
Dc=K2*Vhalf;
k=pck(Ac,Bc,Cc,Dc);

%  end d2hinfsyn
```

D. Validation of the Ellipsoidal Codes

While the ellipsoidal algorithm itself is very simple, the complexity of the derivation of the gradients and their translation into code was a fertile ground for mistakes. Means were consequently required to validate the results. Two methods were used to test their validity. First of all, Rotea presents results for an example problem in [Ref. 29] where an ellipsoidal algorithm was used to determine a measurement-feedback controller by soliving the generalized $\mathcal{H}_2$ synthesis problem . In coding the generalized mixed $\mathcal{H}_2$ /Hi problem, a generalized $\mathcal{H}_2$ algorithm was first written. These results matched Rotea's data exactly. This $\mathcal{H}_2$ code was then modified slightly to accomodate the $\mathcal{H}_\infty$ constraint. Testing the satisfaction of this constraint on the final output was built directly into the termination criteria for the code. Secondly, each gradient subroutine was verified by calculating the gradient at multiple random points in the search space both using the subroutine and by a brute force differencing method. This comparison was very successful in finding errors in either the coding or in the original derivations.

APPENDIX B:CONTROLLER DESIGN EXAMPLE SCRIPTS

This appendix documents the MATLAB scripts and simulink models that were used for the design and analysis of the autoland controllers in Chapter IV. The model used for both controllers was identical, and a single script prepared both synthesis models in order to ensure that the designs were for identical systems. Consequently several of the sections pertain to both designs. The first section presents the simulink models and the equation of motion function file from which the linearized models were extracted and which were used during the analysis. It also includes the scripts which established the linear synthesis models. The second section includes the scripts used to perform the state-feedback and measurement-feedback design for the $\mathcal{H}_\infty$ controller. The third section includes the scripts which were used to design the mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ controller.

A. NONLINEAR MODELS

1. Equation of Motion

The following equation of motion script was used at the core of all the nonlinear models to perform the calculation of the state derivatives. Since the F-14 design problems were limited to longitudinal axis glideslope tracking problems, only the longitudinal states are considered. The requirement to perform robustness analysis and design also required the inclusion of uncertainty inputs and outputs in the equations of motion. These terms are reflected in the variables `delta_in` and `delta_out`. The stability derivative data was extracted from [Ref. 48] for a

flight condition of 134 KIAS and 11.7 degrees in the power approach configuration. Because the control power derivatives for the DLC were unavailable, it was scaled to produce $0.1g$ of vertical acceleration, and $0.01g$ of horizontal deceleration at a deflection of 0.5 *rad.* True states were calculated internal to the equations of motion, and perturbation states were therefore integrated exterior to the equations of motions. This architecture simplified simulation and trimming, since all signals external to the equations of motions were perturbation states. The trimmed lift and drag coefficients were adjusted from their handbook values because they were slightly inconsistent with the values necessary to trim the model at the specified thrust and weight.

eom4

```
function statedot=eom4(state)

%  determines continuous time state derivative for nonlinear longitudinal
%  equations of motion.
% This version all states read in are the perturbation states

%  flight condition:
g=32.174;                          % fps
m=54000/g;                         % slugs
Iyy=247194;                        % slug-ft^2
S=565 ;                            % ft^2
cbar=9.8;                          % ft
alphaT=0;                          % thrust incidence (radians)
zT=0;                              % thrust couple
V0=134*1.6889;                     % fps
rho=.002376;                       % slug/ft^3
alpha0=11.4/57.3;                  % rad
theta0=alpha0;                     % rad - trim point is level flight
T0=13118;                          % lbs

% F-14 derivatives
CDt=0.37405269;       % trimmed coefficients (these differ from handbook)
CLt=1.49534913;
Cmt=0;
CDU=0;
CLU=0;
CMU=0;
CDA=0.0208*57.3;           % rad^-1
CLA=0.0799*57.3;           % rad^-1
```

```
CMA=-0.0115*57.3;          % rad^-1
CDQ=0;
CLQ=5.45;                  % rad^-1
CMQ=-14.3;                 % rad^-1
CDAD=0;
CLAD=-0.51;                % rad^-1
CMAD=-0.93;                % rad^-1

CDIS=0;
CLIS=0.0141*57.3;          % rad^-1
CMIS=-0.0201*57.3;         % rad^-1

% the DLC control power is scaled to permit .1g at full deflection of .5 radians
CLDLC=0.1*CLt/(0.5);
CDDLC=0.1*CLDLC;
CMDLC=0;

% Unpack perturbation states
du=state(1);
dalpha=state(2);
q=state(3);
dtheta=state(4);
stab=state(5);             % These are perturbation deflections
dT=state(6);
DLC=state(7);              % ditto
delta_in=[state(8);state(9);state(10)];

% Calculate true states
u=(du+cos(alpha0))*V0;
alpha=dalpha+alpha0;
theta=dtheta+theta0;
T=T0+dT;

% Calculate the dynamic pressure
V=u/cos(alpha);
Q=0.5*rho*V^2;

%  following matrix incorporates both fixed constants and rotation from stab
%  axis to body axis
Rwb=[-cos(alpha) sin(alpha) 0;-sin(alpha) -cos(alpha) 0;0 0 1];
Rbw=Rwb';
QT=Q*S*[-cos(alpha) sin(alpha) 0;-sin(alpha) -cos(alpha) 0;0 0 cbar];

%  LHS comprises mass matrix and alphadot aero forces
LHS=diag([m*V0,m*V,Iyy])-QT*(cbar/2/V)*[CDAD CLAD CMAD]'*[0 1 0];

% Determine coupling term (body axes)
Fcouple=m*q*[-sin(alpha)*V; u; 0];

% Determine gravity term (body axes)
Fgrav=m*g*[-sin(theta); cos(theta); 0];
```

```
% Determine thrust term (body axes)
Fthrust=T*[cos(alphaT); sin(alphaT); zT];

%  Build aero forces in stability axes and then rotate to body axes
Trim=[CDt; CLt; 0];
Derv=[CDU*V0 CDA CDQ; CLU*V0 CLA CLQ; CMU*V0 CMA CMQ];
CPower=[CDIS CDDLC; CLIS CLDLC; CMIS CMDLC];

Faero=QT*(Trim+Derv*[du;dalpha;q*cbar/2/V]+CPower*[stab; DLC]+ delta_in);
delta_out=Derv*[du;dalpha;q*cbar/2/V]+CPower*[stab; DLC];

statedot1=LHS\(Fcouple+Fgrav+Fthrust+Faero);

thetadot=q;
gamma=theta-alpha;
wdot=statedot1(2)*V;
hdot=sin(gamma)*V/V0;        % Note that hdot is scaled by velocity

statedot=[statedot1;thetadot; hdot; wdot; V;delta_out];

%  end eom4
```

2. Openloop Simulink Model

The open-loop simulink model from which the synthesis model was extracted is presented in Figures B.1, B.2, and B.3. The model includes the actuator models, as well as the appended integrators on the outputs of interest. As discussed in Chapter IV, additional integrators were required midway through the design process in order to achieve the desired tracking properties. These figures include the additional appended integrators. These models were not used for any simulation, but represented a graphical means of accounting for the paths of various signals through the system.

3. Synthesis Model Construction

The following script performed the function of extracting the linear model and creating the state-space formulations for both the $\mathcal{H}_\infty$ controller design and the mixed design. Beyond the linearization itself, the principal purpose of this script was

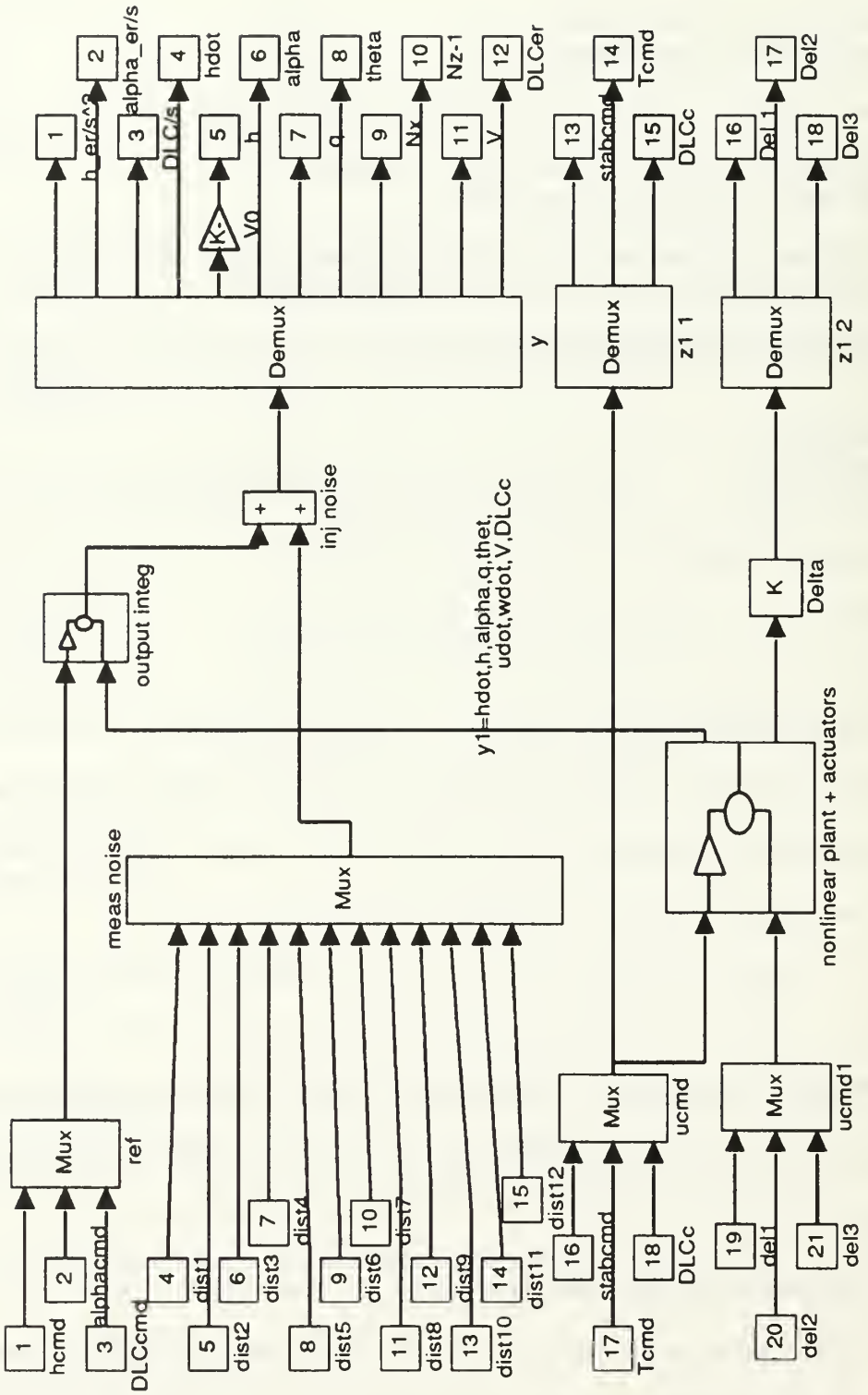

Figure B.1: Open-Loop Simulink Model

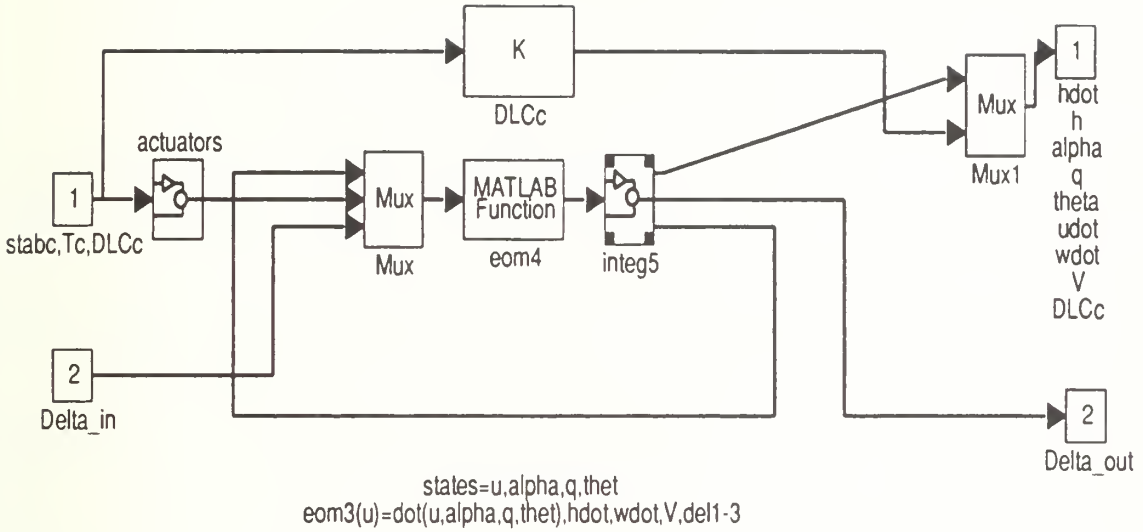

Figure B.2: Non-Linear Plant and Actuators Block

to perform the necessary bookkeeping functions of collecting the appropriate channels into the appropriate inputs and outputs of the vectors w, z_0, z_1, and y. Finally, some baseline scaling of the input and output signals was performed. The system matrices were then compiled in a μ-tools system matrix format [Ref. 24] to be passed to the appropriate design codes.

plant10c

```
% Establish synthesis plant for mixed controller and hinf design problem
% This uses the f14 model extracted from f14nlaero5
% This problem is a glideslope tracker with
% with double integrators on DLC and alpha

Delta=0.2*eye(3);
tau1=0.4;  %  power plant time constant (2.5 rad/s)
tau2=0.05;  %  horizontal stab time constant (20 rad/s)
tau3=0.02;   %  DLC time constant            (50 rad/s)
sigma_gam=3/57.3;  %rad
```

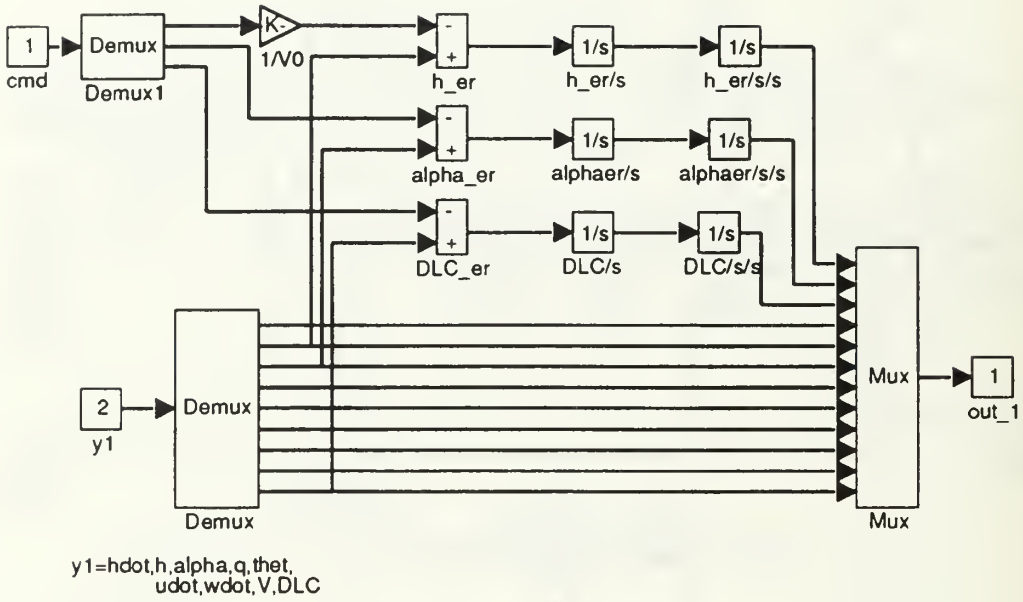

Figure B.3: Output Integrators Block

```
U0=134*1.6889;

[AA,BB,CC,DD]=linmod('f14nlaero5');
% linmod orders states(14):
%     h_er/s2,alphaer/s2,DLC/s2,u,alpha,q,theta,stab,T,DLC,h,alp_er/s,h_er/s,DLC/s
% inputs:(21)
%    h_cmd,alp_cmd,DLC_cmd,13 noises,stabc,Tc,DLCc,del 1-3
% outputs:(18)
%  h_er/s2,alper/s,DCL/s,hdot,h,alpha,q,theta,Nx,(Nz-1),V,DLC,
%     stabcmd,Tcmd,DLCcmd,del1-3

%  Requires plant be input in form:
%
%     xdot= AA*x + B1*w + B2*u
%       z0= C0*x +       + D02*u
%       z1= C1*x +       + D12*u
%        y= C2*x + D21*w



% 'a' represents the full output feedback plant
% 'b' is reduced statefeedback plant (noise columns stripped)
```

```
B1a=BB(:,[1:15,19:21]);     B1b=BB(:,[1:3,19:21]);       B2=BB(:,16:18);

% *************Build Pure Hinf Plant matrices ************************

% Important: delta scaled by factor of .2 already

% z1: h_er/s,alper/s,DCL/s,stabcmd,Tcmd,DLCcmd,udot,alphadot,qdot,thetadot,hdot,
%          del1-3 (14)
C1a=[CC([1:3,13:15],:);  AA([4:7,11],:); CC(16:18,:)];
D11a=zeros(14,18);      D11b=zeros(14,6);
D12a=[DD([1:3,13:15],16:18);  B2([4:7,11],:); DD(16:18,16:18)];

% y: h_er/s2,alper/s2,DCL/s2,h,alpha,q,theta,Nx,(Nz-1),V (10)
C2=CC([1:3,5:11],:);
D21=DD([1:3,5:11],[1:15,19:21]);        D22=DD([1:3,5:11],16:18);

% Pack mixed output feedback plant
pnom_a=pck(AA,[B1a B2],[C1a;C2],[D11a,D12a;D21,D22]);

% Pack statefeedback plant
pnom_b=pck(AA,[B1b B2],C1a,[D11b D12a]);

% To scale outputs for hinf:   z1- rows 14:27
%                              del are rows 25:27, and col 29:31

% ******************** Build Mixed Plant matrices ************************

% Important: delta scaled by factor of .2 already

% z0: h_er/s,alper/s,DCL/s,stabcmd,Tcmd,DLCcmd,udot,alphadot,qdot,thetadot,hdot(11)
C0=[CC([1:3,13:15],:);  AA([4:7,11],:)];
D01c=zeros(11,18);      D01d=zeros(11,6);
D02=[DD([1:3,13:15],16:18);  B2([4:7,11],:)];

% z1:  h_er/s,alper/s,DCL/s,stabcmd,Tcmd,DLCcmd,del1-3 (9)
C1=CC([1:3,13:18],:);
D11c=zeros(9,18);  D11d=zeros(9,6);
D12=DD([1:3,13:18],16:18);

% Pack mixed output feedback plant
pnom_c=pck(AA,[B1a B2],[C0;C1;C2],[D01c D02;D11c,D12;D21,D22]);

% Pack statefeedback plant
pnom_d=pck(AA,[B1b B2],[C0;C1],[D01d D02;D11d,D12]);

% To scale mixed outputs:  z0- rows 15:25;  z1- rows 26:34
% To scale inputs:   noise cols 18:29

% Scale Nominal Mixed Plants:
Scale1=diag([0.001,0.001,0.001,0.001,0.000001,0.001,2*ones(1,3)]);
pnom_c(26:34,:)=Scale1*pnom_c(26:34,:);
```

```
pnom_d(26:34,:)=Scale1*pnom_d(26:34,:);

% Scale w
pnom_c(:,15:17)=0.01*pnom_c(:,15:17);
pnom_d(:,15:17)=0.01*pnom_d(:,15:17);

sdof=14*15/2+3*14;
Dimd=[14 3 6 9 sdof];
Dimc=[14 3 18 9 10 sdof];

Sf=zeros(row(C0));

% end plant10c
```

4. Closed-Loop Analysis Models

Figure B.4 depicts the simulink model used both for the nonlinear simulation and the robustness analysis. A $\mathcal{D}$ implementation of the various controllers was used and can be observed in Figures B.5 and B.6, where the measurement channels are differentiated prior to the controller, and then the output of the controller passes through double integrators.

B. $\mathcal{H}_\infty$ DESIGN SCRIPTS

The following scripts performed the $\mathcal{H}_\infty$ design process including the appropriate scaling of the weighting functions, calling the design code itself, and then analyzing the resulting system. The two subroutines `sfbanal5` and `snsrloop` performed the analysis of the state-feedback system and the analysis of the sensor responses.

loopshape5

```
%  Control Design and analysis for plant10c
%  Plant10c is extracted from f14nlaero5. Includes double integrators
%  on altitude, alpha, and DLC to track altitude and washout alpha and DLC in
%  response to a ramp altitude input.

%  Broken_loop analysis for output feedback
%
%  Determines pure hinf dynamic controller
```

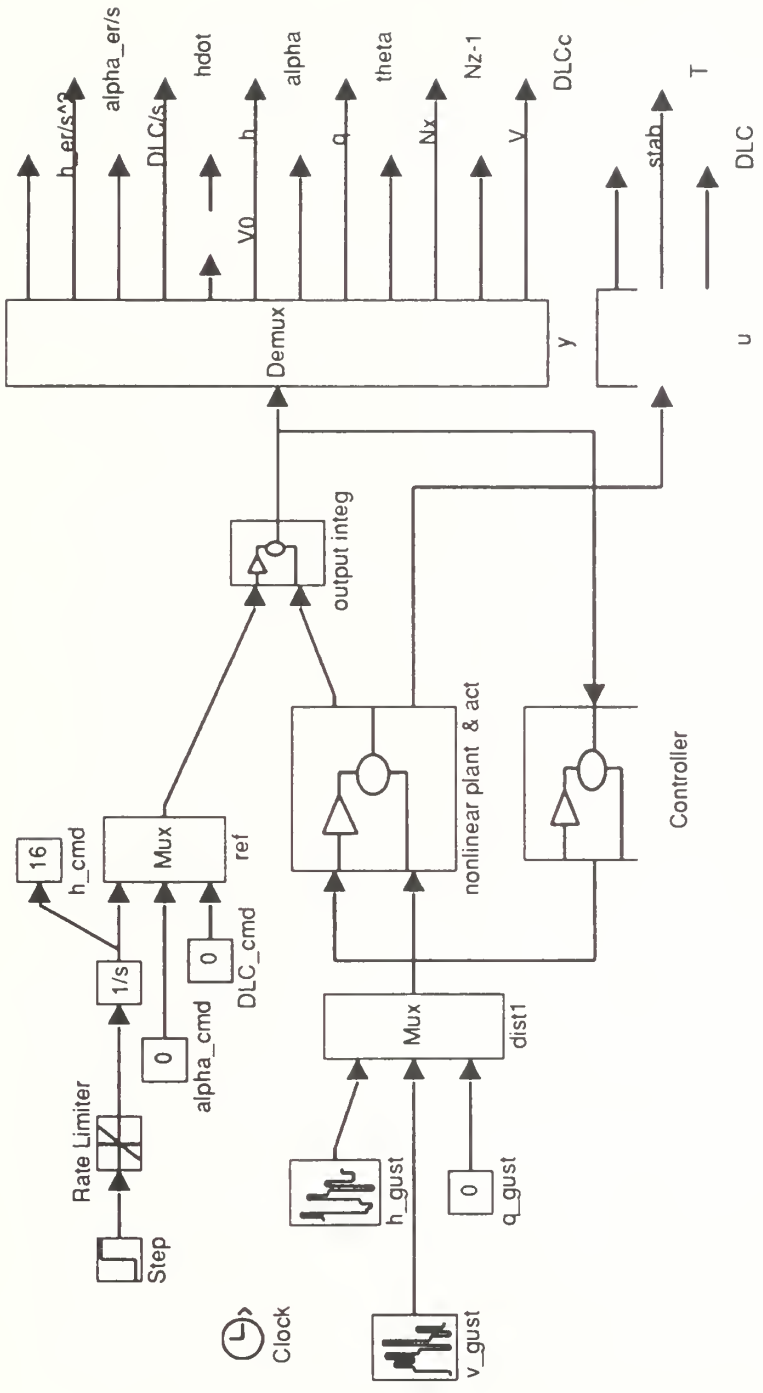

Figure B.4: Closed-Loop Simulink Model

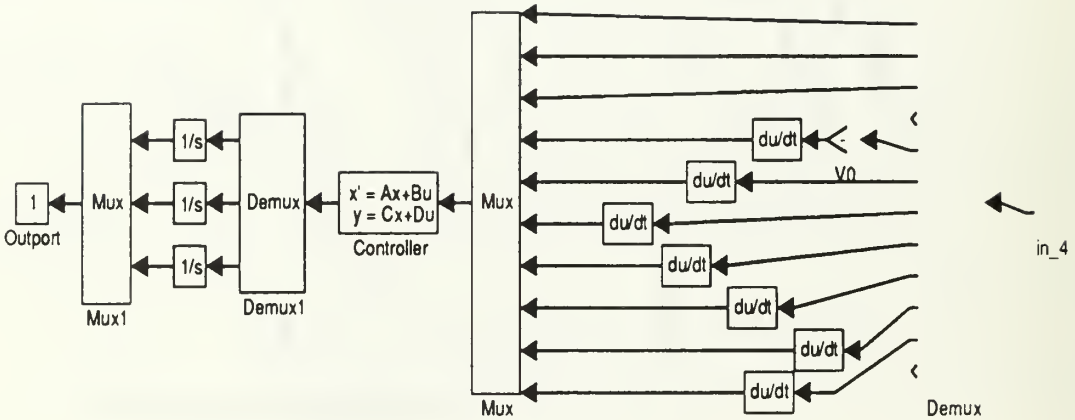

Figure B.5: Controller Block

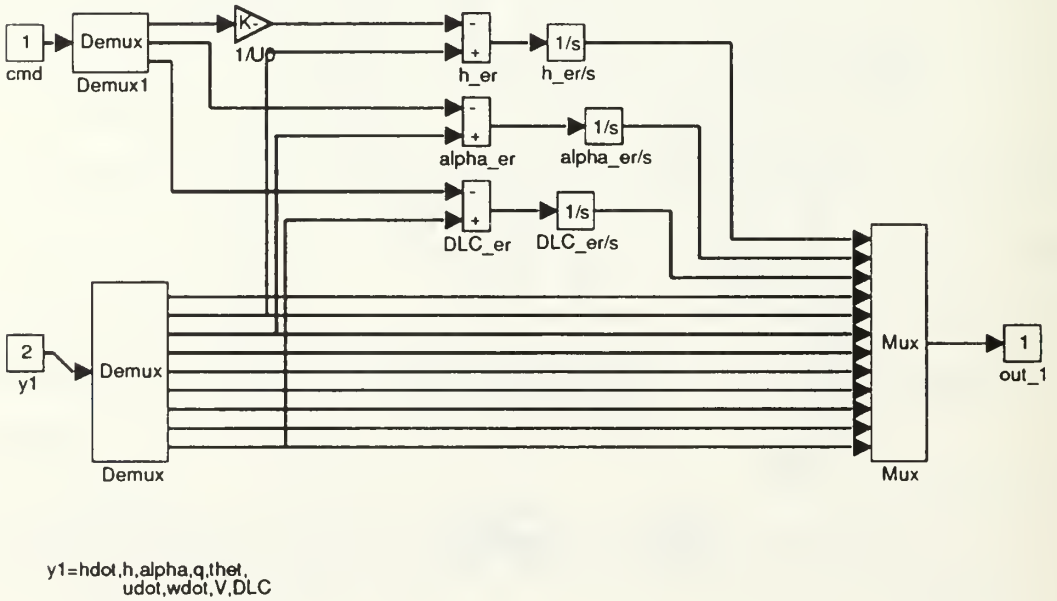

Figure B.6: Output Integrator Block

```
%  Plots brkloop control response vs. sfb design
%  Plots clsdloop command response
%  Plots clsd loop sensor response
 plant10c

% sfb problem weights-- [2 2 .1 5 1 1 0 5 5 5 1 0 0 0]

 sfbanal5, Ksfb=K;

% opfb problem weights
%[p2o,p1]=scale15(pnom_a,pnom_b, [2 5 .5 5 1 5 0 5 5 5 .1 0 0 0]);
[p2o,p1]=scale15(pnom_a,pnom_b,[10 30 5 5 .01 1 0 10 5 1 0 0 0 0]);

distscale=[1,1,1,.00001*[.1 4 4 0 .01 .01 .01 .1 10 10 1 0 5 5 5]];
p2=p2o; p2(:,15:32)=p2o(:,15:32)*diag(distscale);

% calculate hinf opfb controller
K = hinfsyn(p2,10,3,0,50000,500);

[Ac,Bc,Cc,Dc]=unpck(K);
AAcl=[AA B2*Cc;Bc*C2 Ac+Bc*D22*Cc];

 snsrloop4

Bstar=[B2;zeros(size(B2))];
Cstar=[zeros(size(Cc)),Cc];
W=logspace(-3,3,100);

%
% broken loop controller response
%

Ka=Cc;   Ka(1,:)=zeros(1,14);
Tuu1=pck([AA,B2*Ka;Bc*C2,Ac+Bc*D22*Cc],Bstar(:,1),Cstar(1,:),0);
BrkLoop1=frsp(Tuu1,W);

Kb=Cc;  Kb(2,:)=zeros(1,14);
Tuu2=pck([AA,B2*Kb;Bc*C2,Ac+Bc*D22*Cc],Bstar(:,2),Cstar(2,:),0);
BrkLoop2=frsp(Tuu2,W);

Kc=Cc;  Kc(3,:)=zeros(1,14);
Tuu3=pck([AA,B2*Kc;Bc*C2,Ac],Bstar(:,3),Cstar(3,:),0);
BrkLoop3=frsp(Tuu3,W);

figure(7),
vplot('liv,lm',BrkLoop1,'r-',BrkLoop2,'g--',BrkLoop3,'b-.',sfbloops,'y'),grid
title('Broken Loop Controller'),

figure(8),vplot('nyq',BrkLoop1,'r-',BrkLoop2,'g--',BrkLoop3,'b-.',sfbloops,'y'),
grid, title('Nyquist Plot'),
axis([-5 5 -5 5]),axis('square'),axis('equal'),
```

```
%
%  Open Loop controller response
%

Kc=zeros(3,12);
Tuu4=pck([AA,zeros(14,14);Bc*C2,Ac],Bstar,Cstar,zeros(3,3));
Openloop=vsvd(frsp(Tuu4,W));
figure(9),vplot('liv,lm',Openloop),grid,
title('Open-loop Controller Response')

% Closed loop command response
Bin=[BB(:,1:3);zeros(14,3)];
Cout=[CC([5:6,12],:),[zeros(2,14);DD(12,16:18)*Cc]];
Cout2=[CC([5:8,13:15],:),DD([5:8,13:15],16:18)*Cc];

Thh=pck(AAcl,Bin(:,1),Cout(1,:),0);
Tvv=pck(AAcl,Bin(:,2),Cout(2,:),0);
Ttt=pck(AAcl,Bin(:,3),Cout(3,:),0);

Cloop1=frsp(Thh,W);
Cloop2=frsp(Tvv,W);
Cloop3=frsp(Ttt,W);

figure(10),
vplot('liv,lm',Cloop1,'r',Cloop2,'g',Cloop3,'b',sfbcmdloop,'c',dB3thres,'m'),
title('Closed Loop Command Responses'),

% Closed-loop eigenvalues
[V,E]=eig(AAcl); E=diag(E);
disp('Closed Loop eigenvalues and damping ratios')
[E,-cos(angle(E))]

% Open Loop eigenvalues and controller zeros
disp('Open-Loop Poles')
E1=eig(AA)

disp('Controller Zeros')
Kzero=tz(Ac,Bc,Cc,Dc)

%
% mu analysis of the closed-loop systems
%

disp('do you want to do the mu analysis?')
disp('Type qq=0, followed by return to stop, else type return')
qq=1;
keyboard
if qq==0, return, end
W=logspace(-2,3);
```

```
% sfb mu
Asfb=(AA+B2*Ksfb);
Tdel=pck(Asfb,BB(:,19:21),CC(16:18,:),DD(16:18,19:21));  Tdelrsp=frsp(Tdel,W);
MuTsfb=mu(Tdelrsp,[1 1;1 1;1 1]);

% opfb mu
U0=134*1.6889;
Delta=0.2*eye(3);
[Ad,Bd,Cd,Dd]=linmod('f14nlclsd5');
Tdel2=pck(Ad,Bd(:,16:18),Cd(16:18,:),Dd(16:18,16:18));
Tdel2rsp=frsp(Tdel2,W);
MuTopfb=mu(Tdel2rsp,[1 1;1 1;1 1]);
figure(11),vplot('liv,lm',MuTsfb,'-',MuTopfb,'--'),
% title('Structured Singular Values'),
grid
```

sfbanal5

```
% Scale output vector z1
% [p2,p1]=scale15(pnom_a,pnom_b,[2 2 .1 5 1 1 0 5 5 5 1 0 0 0]);
[p2,p1]=scale15(pnom_a,pnom_b,[10 30 5 5 .01 1 0 10 5 1 0 0 0 0]);

% zero out uncertainty columns
p1(:,17:19)=zeros(size(p1(:,17:19)));

% Analysis of state feedback control system
 K = hinffi(p1,3,0,50000,1000);

nn =80;
W=logspace(-3,3,nn);
K=K(:,1:14);

% broken loop controller response

Ka=K;  Ka(1,:)=zeros(1,14);
Tuu1=pck(AA+B2*Ka,B2(:,1),K(1,:),0);
BrkLoop1=frsp(Tuu1,W);

Kb=K;  Kb(2,:)=zeros(1,14);
Tuu2=pck(AA+B2*Kb,B2(:,2),K(2,:),0);
BrkLoop2=frsp(Tuu2,W);

Kc=K;  Kc(3,:)=zeros(1,14);
Tuu3=pck(AA+B2*Kc,B2(:,3),K(3,:),0);
BrkLoop3=frsp(Tuu3,W);

sfbloops=[BrkLoop1(1:nn,1),BrkLoop2(1:nn,1),BrkLoop3(1:nn,1)];
sfbloops=vpck(sfbloops,getiv(BrkLoop1));
```

```
figure(1),vplot('liv,lm',BrkLoop1,'r-',BrkLoop2,'g--',BrkLoop3,'b-.')
title('Broken Loop Controller Response'),grid
figure(3),vplot('nyq',BrkLoop1,'r-',BrkLoop2,'g--',BrkLoop3,'b-.'),
axis([-5 5 -5 5]),grid

% closed loop command response

Thh=pck(AA+B2*K,B1b(:,1),CC(5,:)+DD(5,16:18)*K,DD(5,1));
ClsdLoop4=frsp(Thh,W);

Tvv=pck(AA+B2*K,B1b(:,2),CC(6,:)+DD(6,16:18)*K,DD(6,2));
ClsdLoop5=frsp(Tvv,W);

Ttt=pck(AA+B2*K,B1b(:,3),CC(12,:)+DD(12,16:18)*K,DD(12,3));
ClsdLoop6=frsp(Ttt,W);

sfbcmdloop=[ClsdLoop4(1:nn,1),ClsdLoop5(1:nn,1),ClsdLoop6(1:nn,1)];
sfbcmdloop=vpck(sfbcmdloop,getiv(ClsdLoop6));

dB3thres=cos(pi/4)*ones(size(W));

figure(2),vplot('liv,lm',ClsdLoop4,'r-',ClsdLoop5,'g--',ClsdLoop6,'b-.',dB3thres,'y:')
title('Closed Loop Command Response'),grid


[V,E]=eig(AA+B2*K); E=diag(E); V=abs(V);
[E,-cos(angle(E))]

return
```

snsrloop4

```
%  Analysis of the sensor responses
%  Intended for call by loopshape script

B1q=[zeros(size(Bc));Bc];
CCq=[zeros(size(CC)),CC];

WW=logspace(-5,5,100);

% closed loop sensor response

snsr1=pck(Ac,Bc(:,1),C2(1,:),0);
SnsrLoop1=frsp(snsr1,WW);

snsr2=pck(Ac,Bc(:,2),C2(2,:),0);
SnsrLoop2=frsp(snsr2,WW);

snsr3=pck(Ac,Bc(:,3),C2(3,:),0);
```

```
SnsrLoop3=frsp(snsr3,WW);

snsr4=pck(Ac,Bc(:,4),C2(4,:),0);
SnsrLoop4=frsp(snsr4,WW);

snsr5=pck(Ac,Bc(:,5),C2(5,:),0);
SnsrLoop5=frsp(snsr5,WW);

snsr6=pck(Ac,Bc(:,6),C2(6,:),0);
SnsrLoop6=frsp(snsr6,WW);

snsr7=pck(Ac,Bc(:,7),C2(7,:),0);
SnsrLoop7=frsp(snsr7,WW);

snsr8=pck(Ac,Bc(:,8),C2(8,:),0);
SnsrLoop8=frsp(snsr8,WW);

snsr9=pck(Ac,Bc(:,9),C2(9,:),0);
SnsrLoop9=frsp(snsr9,WW);

snsr10=pck(Ac,Bc(:,10),C2(10,:),0);
SnsrLoop10=frsp(snsr10,WW);

figure(4),clg
vplot('liv,lm',SnsrLoop7,'-',SnsrLoop8,'--',SnsrLoop9,'-.',SnsrLoop10,'-'),hold on
vplot('liv,lm',SnsrLoop4,'r-',SnsrLoop5,'g--',SnsrLoop6,'b-.')
vplot('liv,lm',SnsrLoop1,'r-',SnsrLoop2,'g--',SnsrLoop3,'b-.')

return
```

C. MIXED $\mathcal{H}_2$ / $\mathcal{H}_\infty$ CONTROLLER DESIGN SCRIPTS

The script in this section was used to design the mixed controller described in Chapter IV.

msfbanal

```
% Design and Analysis of mixed state-feedback control system
plant10c
p=pnom_d;

%scale z0
z0scale=[1,10000,1,1000,1,.1,0,0,0,0,0];

p(15:25,:)=diag(z0scale)*p(15:25,:);

[K,Kp,X,E,ct,P1,P2,oc,t]= h2infsyn(1,p,Dimd,Sf,25000,0.01);
```

```
Ksfb=K;

nn =80;
W=logspace(-3,3,nn);

% broken loop controller response

Ka=K;  Ka(1,:)=zeros(1,14);
Tuu1=pck(AA+B2*Ka,B2(:,1),K(1,:),0);
BrkLoop1=frsp(Tuu1,W);

Kb=K;  Kb(2,:)=zeros(1,14);
Tuu2=pck(AA+B2*Kb,B2(:,2),K(2,:),0);
BrkLoop2=frsp(Tuu2,W);

Kc=K;  Kc(3,:)=zeros(1,14);
Tuu3=pck(AA+B2*Kc,B2(:,3),K(3,:),0);
BrkLoop3=frsp(Tuu3,W);

sfbloops=[BrkLoop1(1:nn,1),BrkLoop2(1:nn,1),BrkLoop3(1:nn,1)];
sfbloops=vpck(sfbloops,getiv(BrkLoop1));

figure(1),vplot('liv,lm',BrkLoop1,'r-',BrkLoop2,'g--',BrkLoop3,'b-.')
title('Broken Loop Controller Response'),grid
figure(3),vplot('nyq',BrkLoop1,'r-',BrkLoop2,'g--',BrkLoop3,'b-.'),
axis([-5 5 -5 5]),grid

% closed loop command response

Thh=pck(AA+B2*K,B1b(:,1),CC(5,:)+DD(5,16:18)*K,0);
ClsdLoop4=frsp(Thh,W);

Tvv=pck(AA+B2*K,B1b(:,2),CC(6,:)+DD(6,16:18)*K,0);
ClsdLoop5=frsp(Tvv,W);

Ttt=pck(AA+B2*K,B1b(:,3),CC(12,:)+DD(12,16:18)*K,0);
ClsdLoop6=frsp(Ttt,W);

sfbcmdloop=[ClsdLoop4(1:nn,1),ClsdLoop5(1:nn,1),ClsdLoop6(1:nn,1)];
sfbcmdloop=vpck(sfbcmdloop,getiv(ClsdLoop6));

dB3thres=cos(pi/4)*ones(size(W));

figure(2),vplot('liv,lm',ClsdLoop4,'r-',ClsdLoop5,'g--',ClsdLoop6,'b-.',dB3thres,'y:')
title('Closed Loop Command Response'),grid

[V,E]=eig(AA+B2*K); E=diag(E); V=abs(V);
[E,-cos(angle(E))]

return
% end msfbanal.m
```

APPENDIX C:INTERIOR POINT CODES

A. GENERAL REMARKS

This appendix documents the interior point codes used during for many of the example problems cited in the body of the report. The codes and their structure are briefly described followed by a verbatim listing of the routines. The codes are followed by comments on practical issues of their use. The original codes upon which these are based were provided by Professor Khargonekar of the University of Michigan and were the result of work performed by Enrique Bayens, his student. The algorithm follows [Ref. 14] as outlined in Chapter II, and is written as MATLAB function files.

These codes are problem independent, and are structured to solve the Eigenvalue Problem (EVP), or the more general Generalized Eigenvalue Problem (GEVP). Recall that the GEVP has the form:

$$\begin{aligned} &\text{Minimize: } \lambda, \\ &\text{Subject to: } \begin{bmatrix} \lambda B(x) - A(x) & 0 & 0 \\ 0 & B(x) & 0 \\ 0 & 0 & C(x) \end{bmatrix} > 0, \end{aligned}$$

where $A(x) = A_0 + \sum_{i=1}^{n} x_i A_i$, $B(x) = B_0 + \sum_{i=1}^{n} x_i B_i$, and $C(x) = C_0 + \sum_{i=1}^{n} x_i C_i$. The EVP is the simplified case where $B(x) = I$. We will refer to the set $\{A_0, A_1, \ldots, A_n\}$ as the *basis* for the affine matrix functional $A(x)$. The application of these codes to any particular problem simply requires that the problem be posed as a GEVP and the three sets of bases determined and stored.

1. Posing a Basis

Two important structural concepts introduced in Bayens' original code provide a mechanism for passing these bases to the interior point routines. The first

is what he calls an "a-matrix", which is simply an accounting method for storing a basis as a single argument, which can easily be passed from function to function in MATLAB's workspace. For example if $A(x) = A_0 + \sum_{i=1}^{n} x_i A_i$, then the corresponding a-matrix is `Aa`$= [A_0, A_1, \ldots, A_n]$. For many control problems, these matrices can be huge, and the interior point methods can consequently be RAM intensive. In many applications, the matrices $A(x)$ and/or $C(x)$ may have structure. That is, $A(x)$ or $C(x)$ may themselves be comprised of diagonal blocks of matrices. In this case, it is advantageous to use the structure to reduce the computational expense. The codes each then ask for a structure matrix `Sc`, which defines the internal structure of a set of basis matrices `Ca`. The number of rows of `S` is then the number of blocks in $C(x)$. Each row has two scalar values. The first value is a "1" or "0", and signifies whether the associated block is a full block, or is itself a diagonal matrix. The second value in that row defines the size of the block, i.e., the number of rows and columns. These bases are then passed to the interior point routines, along with an initial feasible point and any knowledge of the internal structure of $A(x)$ or $C(x)$. Note that for all the problems actually solved in this report, the EVP was considered and so the a-matrix for $B(x)$ was `Ba`$= [I,$ zeros$]$, with sufficient zeros such that the size of `Ba` was identical to the size of the a-matrix `Aa`.

2. Modifications to the Original Code

The codes presented here are structurally identical to the originals, but differ substantially in execution. The architecture of the inputs, outputs and subroutines are unchanged. Most of the changes were in the interest of either numerical efficiency or accuracy. Several changes, for example, took advantage of the structure of many LMI's to reduce the number of multiplications by blocks of zeros.

The most important changes dealt with the way in which matrix divisions were handled. The interior point codes involve frequent divisions by positive definite matrices. The original codes extensively used MATLAB's `inv` function. Replacement with MATLAB's matrix division notation was occasionally satisfactory and efficient, as MATLAB 4.0 first attempts division by Cholesky factorization for any symmetric matrix. Many of the divisions, however, involve positive definite matrices which are very badly conditioned, to which MATLAB responds with a warning message. This is to be expected since one is trying to force an LMI to the boundary only ϵ away from singularity. At the suggestion of Professor Laurent El Ghaoui (ENSTA, Paris), all poorly conditioned divisions were executed by first performing an eigen decomposition of the positive definite denominator. The result is a real diagonal matrix of eigenvalues and an orthonormal matrix of eigenvectors ($H^{-1}L = U\Lambda^{-1}U^TL$, where $H = U\Lambda U^T$). Multiplying by orthonormal matrices and then division of each row by the corresponding scalar eigenvalue optimized the accuracy of the process. Consequently the expression:

```
HL=inv(H)*L;
```

was replaced with:

```
[UH,DH]=eig(H);
L1=UH'*L;
for i=1:order(H),
   L2(i,:)=DH(i,i)\L1(i,:);
end
HL=UH*L2;
```

This is not at all the most efficient means of performing the division, but their badly conditioned nature has led more notable researchers to choose the accuracy implicit in this approach. It is however much faster than allowing MATLAB's matrix division function to wrestle with very poorly conditioned divisions.

3. The Principal Code

The function `centers3`, and its variations is the top level program that solves the EVP. The "3" simply identifies the code as being the third variation to Bayens' original. Three variations to `centers3` were used. The first `centers3b` simply suppressed workspace printing. The next `centers3c` suppressed workspace printing and included the structure of A as an input argument (the original had only used the structure of C). Finally, `centers3d` restored workspace printing, while continuing to permit the structure of both A and C to be passed.

As discussed above, `centers3` is not problem specific, but suitable for solving any EVP. As input arguments it requires the bases (a-matrices) associated with the three EVP matrix functionals, $(A(x), B, C'(x))$; a matrix which defines any internal structure of the problem; and feasible initial conditions x and λ from which to start the search. (Finding a good initial point is discussed later in this appendix). The remaining inputs are numerical thresholds and adjustments. Since `centers3` is not problem specific, it was used for all EVP's, including the mixed discrete and continuous problems, and the plant optimization problems. In each of those cases, scripts were written which prepare the a-matrices appropriate to each problem, and determine a suitable initial conditions before calling `centers3`.

As written by Bayens, the original `centers` was intended to solve the more general GEVP, but a bug in the subroutine `assump2` led me to close the path such

that only constant B matrices were permitted. Since a GEVP solver was not required during this work, the bug was not isolated or corrected. Little work should be required in order to permit the code to solve the GEVP, if it becomes required.

4. The Subroutines

A large number of function files accompany and are called by the central function, `centers3`. Several of these functions could be regarded as administrative, and are used to manipulate a-matrices. They are useful both within `centers3`, and also in preparing a-matrices in problem specific applications prior to calling `centers3`. The functions `affin` and `getvec` use the a-matrix (basis) of A to map back and forth between x and $A(x)$: $A(x) = \text{affin}(Aa, x)$ and $x = \text{getvec}(Aa, A(x))$. The function `adiag` manipulated a-matrices of $D(x)$ and $E(x)$ to find the a-matrix of the functional $F(x) = \begin{bmatrix} D(x) & 0 \\ 0 & E(x) \end{bmatrix}$. This was used extensively whenever two LMI's were jointly imposed.

The following set of functions could rightly be called subroutines, as their use is restricted to calls from within `centers3`. The functions `assump1b`, `assump2`, and `assump3c` checked the necessary assumptions before starting the method of centers. First, `assump1b` verified that the initial point is feasible. Next, `assump2` confirmed that $B(x)$ was bounded away from singular for the GEVP. It currently not called by centers3, and would have to be fixed in order to solve the GEVP. It is included here for completeness. The subroutine `assump3c` actually performed the first iteration of the search for a analytical center in order to determine if the problem is bounded. If this first iteration converged, then the problem was considered bounded. The majority of the work was performed by the subroutine `nesnem3a`, which actually performed the search for the analytic center within the inner-loop of

the method of centers. Within its loop, it in turn called `grad2b`, which returned the gradient and hessian at each step of the search.

B. PRACTICAL ISSUES

This section addresses a number of practical issues regarding the use of these codes. The first of these comments is relevant to the use of any interior point codes for the solution of LMI's. Subsequent remarks pertain specifically to the author's experience in the use of this particular implementation.

1. Determining a Feasible Initial Point

All the interior point methods are dependent upon initializing the optimization with $x^{(0)}$ and $\lambda^{(0)}$ in the feasible set, i.e. $\lambda^{(0)}I - A(x^{(0)}) > 0$ and $C(x^{(0)})$. In fact, the codes in section C verified this requirement prior to any optimization attempt. As a matter of practical experience, it is not just enough to initialize the algorithm at any feasible point, but ideally a point well away from the boundary for which the problem becomes singular (unfeasible). Riccati methods are a very poor choice for determining a feasible initialization point. This is because the numerical methods for solving the Riccati equation must generally make allowance for a tiny negative eigenvalue in an otherwise positive definite solution. Even a very tiny negative eigenvalue however places the corresponding vector x outside boundary function of the LMI with no means of crossing the boundary. One method, suggested by El Ghaoui, is to perform a preparatory optimization by setting $\hat{C}(x) = 1 > 0$, $x = [0, \ldots, 0]^T$, and minimizing $\hat{\lambda}$ such that $\hat{\lambda}I + C(x) > 0$. For the original problem to be feasible, a $\hat{\lambda} < 0$ must exist. Furthermore, by minimizing $\hat{\lambda} < 0$ (finding its maximum negative absolute value), the "best" feasible value of $x^{(0)}$ is determined. The problem of interest can then be initialized with $x^{(0)}$ and some $\lambda^{(0)} >> \lambda_{max}(A(x^{(0)})$.

A modification to this approach was routinely used. Specifically $C(x)$ usually had some structure such as $C(x) = \begin{bmatrix} R(x) & \\ & Y(x) \end{bmatrix}$, and the first n elements of x were the diagonal elements of Y. In this case the preparatory problem was posed as minimize $\tilde{\lambda}$ such that $\tilde{\lambda} I + R(x) > 0$, subject to $Y(x) > 0$. This problem is easily feasibly initialized at $x^{(00)} = [1, 1, \ldots, 1, 0, \ldots, 0]^T$, such that $Y(x^{(00)}) = I > 0$, and with $\lambda^{(00)} >> \lambda_{max}(R(x^{(00)}))$. Consequently, the solution of an LMI frequently required two passes through the interior point algorithm; the first to determine a feasible solution, and the second to determine the (sub)optimal solution.

2. Practical Observations

The following observations were made relative to this specific implementation of the interior point method.

1. Much of the clamor surrounding interior point methods is due to their reputed dramatic increase in computational speed over other convex methods. This makes intuitive sense in that the method does not spend any time outside of the feasible set, while we found it common for 90% of the iterations of the Ellipsoidal methods to be spent isolating the feasible set. In practice, we found little difference in speed between the two methods on small problems. I am certain this is because of the choice of MATLAB as the programming language. MATLAB was chosen because that was language of the original codes provided by Khargonekar, and because it could be rapidly implemented for the desired problems. Computational speed was not as critical as speed of implementation. With the attention the interior point methods have received in the past two years, translating these codes to a faster format would be unadvisable, as commercial codes are sure to become available shortly.

2. Mathematically, the interior point codes are guaranteed to stay within the feasible set due to the influence of the boundary function [Ref. 15]. This was not our practical experience. Two factors routinely led to the method of centers wandering out of the feasible set. First of all, the numerical adjustment θ determines how close to singular the search for an analytical center is initialized. Mathematically, the code should converge for all $\theta \in [0, 1]$. In our experience, values much less than 0.1 resulted in the path of centers occasionally jumping outside the feasible set. Secondly, in a similar vein, the method should work if initialized at any $\lambda^{(0)} > \lambda_{max}(A(x^{(0)})$. Again, if the problem was initialized too close to the boundary for which the LMI was singular, then path of centers could wander infeasible before converging to a optimum value. Consequently, $\lambda^{(0)}$ was usually chosen well clear of the boundary, on the order of 0.1 or 1 greater $\lambda_{max}(A(x^{(0)})$. Furthermore, there were some problem geometries for which the codes were inexplicably unstable. Traps were implemented in the code in order to rapidly identify divergence from the feasible set, thereby slowing the general execution of the routines. This was done in order to alert the researcher to bad output, but did not correct the problem. The source of these difficulties is unquestionably the inherently ill-conditioned nature of these problems, their very intent being to puch the problem to singularity. Again, the advent of more robust commercial codes would hopefully mitigate these sensitivities.

C. MATLAB FUNCTION FILES

This section includes a listing of the interior point optimization code `centers3` and its associated subroutines.

centers3

```
function [xopt,lambdaopt,stat,err]=centers3(A,B,C,Sc,x,lambda,theta,prec,bound)
%  Function:
%  [xopt,lambdaopt,stat,err]=centers3(A,B,C,Sc,x,lambda,theta,prec,bound)
%
% Description:
%
% This function solves the problem of minimize the maximum generalized
% eigenvalue of the pair (A(x),B(x)) subject to a constraint C(x)>0.
% A(x) and B(x) are a (symmetric, symetric-positive-definite) pair of
% matrices that depend affinely on a vector-valued variable x.
%
% Inputs:
%
% A=[A0,A1,...,An],
% B=[B0,B1,...,Bn],
% C=[C0,C1,...,Cn], are a-matrices, that is arrays of matrices that
%    represent an affin matrix expression depending of a vector x.
% Sc is a matrix representing the block structure of the restriction
%    C(x)>0. Sc has 2 columns and the number of rows equals the number
%    of blocks in C(x). The first column contains the type of block
%    (type 0 diagonal block, type 1 full block), the second column is
%    The size of each block..
% x=[x1,x2,...,xn]', is an initial feasible vector.
% lambda is an initial value which satisfies lambda*B(x)-A(x)>0.
% theta is a parameter with 0<theta<1. Typical values are close to 0.
% prec is the precision in computing the optimun.
% bound is a limit of norm(x) to detect unboundness of the
%    problem.
% The affin matrix functions can be calculated using the function affin
% A(x)=affin(A,x)=A0+A1*x1+...+An*xn,
% B(x)=affin(B,x)=B0+B1*x1+...+Bn*xn,
% C(x)=affin(C,x)=C0+C1*x1+...+Cn*xn,
%
% Outputs:
%
% xopt is the optimal vector
% lambdaopt is the maximum generalized eigenvalue
% stat is a matrix with the statistcs of the algorithm. The first column
%    contains the iteration number, the second the number of Newton_NN
%    iterations, and the third a bound of the error in the compute of the
%    optimal value lambdaopt.
%
% Necessary assumptions to solve the problem:
%
% (1) The initial point is in the feasible set.
% (2) B is bounded away from singular on the feasible set, and we know
%     bmin such that B(x) > bmin*I
% (3) The feasible set is bounded.
% err is an error code. If err=1 the initial point is not feasible, if
% err=2 assumption (2) does not hold, if err=3 the problem is unbounded,
% in this case xopt is yet a feasible point, but norm(xopt) >= bound(1 +
```

```
% norm(x)).
%
% Called functions:
%
% assump1, checks assumption (1).
% assump2, checks assumption (2).
% assump3c, checks assumption (3), uses eigen division with Hessian
% bound1, computes a bound of the difference between lambda and the real
%    minimal value
% affin, computes an affin matrix function.
% adiag, computes the a-matrix of C(x) where C(x)=diag[A(x),B(x)].
% nesnem3a, Newton algorithm to compute an analytic center.(eigen division)
% grad2b   returns barrier gradient and hessian (structured version)
%
% As of 1530 on 10/12/93 this is the best performing combination so far.
%   Direct division by hessian leads to enormous increase in cputime,
%   probably because matlab does svd division when it sees an ill-
%   conditioned problem.

%
% Checking assumption (1)
%
disp('Method of centers. Checking initial point ...');
err=assump1b(A,B,C,x,lambda);
if sum(err')<0
   disp('Error in centers.m. Infeasible initial point');
   disp(err)
   return;
end
%
% Checking assumption (2)
%
disp('Method of centers. Checking if B(x) is bounded ...');
[mb,nb]=size(B);
if max(max(abs(B(:,mb+1:nb))))==0,    %is B(x) constant?
   bmin=min(eig(B(:,1:mb)));
   constantb=1;
else,
%   bmin=assump2(A,B,C,x,bound);  % I THINK THIS PATH HAS A BUG!
   disp('non-constant b is currently disabled')
   constantb=0;
end
if bmin<=0
   disp('Error in centers.m. B(x) is not bounded away of zero');
   err=2;      return;
end
%
% Checking assumption (3)
%
disp('Method of centers. Checking if restriction is  bounded ...');
[err,x2,lambda3,Z,it2]=assump3c(A,B,C,Sc,x,lambda,theta,bound);
```

```
if err==3
   disp('Error in centers.m. The restriction is unbounded');
   xopt=x2;
   lambdaopt=lambda3;
   return;
end
%
% Algorithm of the centers
%
[na,ma]=size(A);       [mz,nz]=size(Z);    S=[1,na;Sc];
it1=1;
cond=mz/bmin/trace(Z(1:na,1:na));
Ax=affin(A,x2);              Bx=affin(B,x2);
lambda2=max(eig(Ax,Bx));
stat=[1,it2,cond,lambda2];
disp(stat);
cond=1;      % resets cond to force second pass through nesnem

while(cond>prec)
   x1=x2;
   lambda1=lambda3;
   lambda3=(1-theta)*lambda2+theta*lambda1;
   if constantb==1,
      F=adiag(-A,C);
      F(1:na,1:na)=F(1:na,1:na)+lambda3*B(1:na,1:na);
      [x2,Z,it2,mf]=nesnem3a(F,x1,S);
      Ax=affin(A,x2);
      lambda2=max(eig(Ax));
   else,
      F=adiag(lambda3*B-A,C);
      [x2,Z,it2,mf]=nesnem3a(F,x1,S);
      Ax=affin(A,x2);           Bx=affin(B,x2);
      lambda2=max(eig(Ax,Bx));
   end
   cond=mf/bmin/trace(Z(1:na,1:na));
   it1=it1+1;
   stat1=[it1,it2,cond,lambda2];
   if lambda2>lambda3,
       error('centers3 blowing up')
   end
   stat=[stat;stat1];
   disp(stat1);
end
xopt=x2;
lambdaopt=lambda2;
tol=cond;

return;

% end centers3
```

assump1b

```
function err=assump1b(A,B,C,x,lambda)
%
% err=assump1(A,B,C,x,lambda)
%
% Description:
%
% This function checks that the initial point for the algorithm of the
% centers is admissible.
%
% Inputs:
%
% A a-matrix
% B a-matrix
% C a-matrix
% x vector compatible with the a-matrices.
% lambda
%
% Outputs:
%
% err is an error code, err=[0 0 0] if the initial point is admissible, err<0
%     otherwise.
%
% Comments:
%
% See function centers
%
err=[0 0 0];
z1=min(eig(lambda*affin(B,x)-affin(A,x)));
z2=min(eig(affin(B,x)));
z3=min(eig(affin(C,x)));
if z1<=0
   err(1)=z1;
end
if z2<=0
   err(2)=z2;
end
if z3<=0,
   err(3)=z3;
end
return;

% end assump1b
```

assump2

```
function bmin=assump2(A,B,C,x,bound)
%
```

```
% bmin=assump2(A,B,C,x,bound)
%
% Description:
%
% This function check that the restriction B(x)>0 is bounded away from
% be singular, it determines a lower bound bmin solving an auxiliary
% minimization problem. If B(x)=B constant matrix, bmin is the minimum
% eigenvalue of B.
%
% Inputs:
%
% A a-matrix,
% B a-matrix,
% C a-matrix,
% x vector,
% bound is a bound to detect that C(x) is unbounded.
%
% Outputs:
%
% bmin is a bound for B(x)
%
% Comments:
%
% See function centers
%
%
[na,ma]=size(A);
%
% If B(x) is a function of x
%
if B(:,na+1:ma)~=zeros(na,ma-na)
   A1=-B;
   A2=aident(size(A1));
   lambda=max(eig(affin(A1,x)))+1;
   [x1opt,b,stat,err]=centers(A1,A2,C,Sc,x,lambda,theta,prec,bound);
   if err==0
      bmin=-b;
   else
      error('Error checking assumption (2)');
   end
%
% If B(x)=B constant matrix
%
else
   bmin=min(eig(B(:,1:na)));
end
return;
```

assump3c

```
function [err,xnew,lnew,Z,it]=assump3c(A,B,C,Sc,x,lambda,theta,bound)
%
% [err,xnew,lnew,Z,it]=assump3b(A,B,C,Sc,x,lambda,theta,bound)
%
% This function checks assumption (3) of the method of centers using
% one iteration of the centers method, if the norm of x diverges, then
% an unbounded direction exists. It returns the analytic center x,  the
% value of the maximum eigenvalue lnew, and an error code which is err=3
% if the problem is unbounded and err=0 otherwise.
%
% Inputs:
%
% A,B,C a-matrices.
% Sc matrix with the block structure of C(x)
% x vector
% lambda initial value
% theta parameter of the method of centers.
% bound is a limit for the norm of x to detect divergence.
%
% Outputs:
%
% err error code. err=3 if the problem is unbounded, otherwise err=0.
% xnew analytic center.
% lnew new value of lambda for xnew
% Z=inv(F(x))
% it is the number of iterations
%
% Comments:
%
% See function centers.
% Modified to call grad2b, uses eigen division

[na,ma]=size(A);
Ax=affin(A,x);
Bx=affin(B,x);
lambda1=max(real(eig(Ax,Bx)));
lnew=(1-theta)*lambda1+theta*lambda;
F=adiag(lnew*B-A,C);
S=[1,na;Sc];
x2=x;           delta2=1;             it=0;
while((delta2>0.001)&(norm(x2)<bound))
   x1=x2;
   delta1=delta2;
   [g,H,Fx,mf]=grad2b(F,x1,S);
   [uH,dH]=eig(H);
   if any(diag(dH)==0),
      error('Hessian collapsed trying to analyze assump3'),
   end
   Hg=uH*(diag(dH).\(uH'*g));
   delta2=sqrt(g'*Hg);
   if delta2 > .25
```

```
      alpha=1/(1+delta2);
   else
      alpha=1;
   end
   x2=x1-alpha*Hg;
   it=it+1;
end
[uF,eF]=eig(Fx);
Z=uF*(eF\uF');
if any(diag(eF)<0)
   error('Fx is not positive definite in assump3c')
end
xnew=x2;
Fx2=affin(F,x2);
if any(eig(Fx2)<=0)
   disp('Terminal point in assump3c infeasible- reverted to previous')
   xnew=x1;
end
if norm(x2)>bound
   err=3;
else
   err=0;
end
return;
% end assump3c
```

affin

```
function F = affin(A,x,col)
%
% F = affin(A,x,col)
%
% Description:
%
% This function computes the affine matrix expression
% F = A0 + A1 x1 + ... + Ar xr. If A is an empty a-matrix,
% the function returns F=[].
%
% Inputs:
%
% A is an a-matrix A=[A0,A1,...,Ar],
% x is a vector x=[x1,...,xr].
% col is the number of columns of A, if not provided, it is assumed
%     equal to the number of rows.
%
% Outputs:
%
% F is the matrix F = A0 + A1 x1 + ... + Ar xr.
%
```

```
if A==[]
   F=[];
   return;
else
   [ma,na]=size(A);
   [mx,nx]=size(x);
   if nargin==2
      col=ma;
   end
   n=na/col;                    % number of matrices in A
   if n~=max(mx,nx)+1
      error('Error in affin.m. Incompatible dimensions')
   end
   F=A(:,1:col);
   for i=1:n-1
      j=col*i;
      F=F+A(:,j+1:j+col)*x(i);
   end
end
return;
```

adiag

```
function F=adiag(B,C)

% given the a-matrices B and C, finds the a-matrix F such that
% affin(F,x)=[affin(B,x) ,0; 0, affin(C,x)]
%
% calls daug from mutools

[nb,mb]=size(B);   [nc,mc]=size(C);
dim=mb/nb;
F=[];
for i=0:dim-1,
   k1=i*nb;     k2=i*nc;
   F=[F,daug(B(:,k1+1:k1+nb),C(:,k2+1:k2+nc))];
end

% end adiag
```

nesnem3a

```
function [xopt,Z,iter,mf]=nesnem3a(F,x,S)
%
% [xopt,Z,iter]=nesnem3(F,x,S)
%
```

```
% Description:
%
% This function determines the analytic center 'xopt' which minimizes
% the barrier function f(x)=log det inv(F(x)) using Nesterov and
% Nemirovsky's Newton algorithm, where F(x)=F0+F1*x1+...+Fr*xr.
%
% Inputs:
%
% F = [F0,F1,...,Fn],
% x = [x1,...,xn], is a initial value,
%
% Outputs:
%
% xopt analytic center
% Z=inv(F(x)) which will be used later in the stopping criteria
%    for the method of the centers, actually is calculated in function
%    "grad".
% iter number of iteration to find the analytic center.
%
% Related functions:
%
% grad2b computes gradient and hessian of f(x)=log det inv(F(x)).
%
% Comments:
% Called by centers.
%
% This version uses eigen decomposition for ill-conditioned divide.  Computes Z out of loop

x1=x;     iter=0;      delta=1;
while(delta>.001)
   [g,H,Fx,mf]=grad2b(F,x1,S);   % Determination of gradient and hessian
   [uH,dH]=eig(H);
   if any(diag(dH)==0),
      disp('Hessian collapsed'),
      break
   end
   Hg=uH*(diag(dH).\(uH'*g));
   delta=sqrt(g'*Hg);             % Newton decrement
   if delta > .25
      alpha=1/(1+delta);
   else
      alpha=1;
   end
   x1=x1-alpha*Hg;
   iter=iter+1;
end
xopt=x1;      Z=inv(Fx);
return;
% end nesnem3a
```

```
function [g, H, Fx, mf] = grad2b(F,x,S)
%
% [g, H, Fx] = grad2(F,x,S)
%
% Description:
%
% This function determines the gradient 'g' and the hessian 'H'
% of the barrier function f(x) = log det inv(F(x)),
% gi(x)=-Trace(ZFi), i=1,...,n,
% Hij(x)=Trace(ZFi)(ZFj), i=1,...,n,
% Z=inv(F(x))
%
% Inputs:
%
% F = [F0,F1,...,Fn],
% x = [x1,...,xn],
% S is the block structure of F(x), S=[s1 s2], where s1 is a vector
%   with the type of block (type=0 is a diagonal block, type=1 is a full
%   order block), s2 is a vector with the size of each block. If does not
%   exist matrix S, it is supposed that F(x) is a complete full order
%   block.
%
% Outputs:
%
% g gradient
% H hessian
% Fx= F(x)
%
% This version differs from the original by taking advantage
% of MATLAB4's smart division, and differs from grad2 by exploiting
% structure.
% Significant flops are saved by taking advantage of the structure
% of the argument of the trace in computing the hessian.
% Block division is desirable since we're forcing an ill-conditioned
% problem, and the smaller the blocks, the more accurate the division
% will be. F(x) should be PD, so matlab4 will use cholesky as long as
% the subblocks of F(x) remain numerically PD.
%
% Comments:
%
% See functions nesnem and centers.
%
[mf,nf]=size(F);
if nargin==2,
   S=[1,mf];
end
[ms,ns]=size(S);
dim=nf/mf-1;
Fx=affin(F,x);
```

```
g=zeros(dim,1);          H=zeros(dim,dim);          ind1=1;
for j=1:ms,
   if (S(j,1)==0) | (S(j,2)==1),
      for i=1:S(j,2),
         FxF=Fx(ind1,ind1)\F(ind1,ind1+mf:mf:ind1+dim*mf);
         g=g-FxF';
         H=H+FxF'*FxF;
         ind1=ind1+1;
      end
   else,
      size=S(j,2);        k1=ind1:ind1-1+size;       k2=zeros(1,size*dim);
      for i=1:dim,
         k2(1+(i-1)*size:i*size)=k1+i*mf;
      end
      FxF=Fx(k1,k1)\F(k1,k2);
      for i=1:dim,
         l1=FxF(:,size*(i-1)+1:size*(i-1)+size);
         g(i)=g(i)-trace(l1);
         H(i,i)=H(i,i)+sum(sum(l1.*l1'));
         for k=i+1:dim,
            l2=FxF(:,size*(k-1)+1:size*(k-1)+size);
            H(i,k)=H(i,k)+sum(sum(l1.*l2'));
            H(k,i)=H(i,k);
         end
      end
      ind1=ind1+size;
   end
end
return;

% end grad2
```

APPENDIX D:PLANT/CONTROLLER OPTIMIZATION CODES

This appendix documents the MATLAB script and function files which were used in solving the example problems discussed in Chapter V. The first section is the function files which perform the optimization. These functions are independent of the specific example to be solved, but instead solve a general class of problem, such as the joint $\mathcal{H}_\infty$ pole placement problem. The second section is the scripts which were used to prepare the synthesis models for the example problems. The outputs of these scripts were used as the input arguments for the optimization functions in the first section.

A. PLANT/CONTROLLER OPTIMIZATION FUNCTIONS

This section documents the function files which perform the plant controller optimization process. These functions all had similar structure, and input and output arguments. The input arguments included the synthesis model, initial plant and cost vector , and the optimal plant and associated controller were outputs. The synthesis models were required to be state-space representations of the plants with A, B_1 and B_2 expressed as "a-matrices" of the associated plant variables. That is if:

$$\begin{aligned} A &= A(\zeta) = A_0 + \sum_{i=1}^{r} \zeta_i A_i, \\ B_1 &= B(\zeta) = B_{1_0} + \sum_{i=1}^{r} \zeta_i B_{1_i}, \\ B_2 &= B(\zeta) = B_{2_0} + \sum_{i=1}^{r} \zeta_i B_{2_i}, \end{aligned}$$

then the associated a-matrices were Aa= $[A_0, A_1, \ldots, A_r]$, B1a= $[B_{1_0}, B_{1_1}, \ldots, B_{1_r}]$, and B2a= $[B_{2_0}, B_{2_1}, \ldots, B_{2_r}]$. The matrices Aa, B1a, and B2a are then the *bases* for the plant matrices A, B_1, and B_2.

Per Chapter V, the general method was to first find a feasible controller which minimized the controller margin for the initial plant (maximized its absolute value), then freeze the controller while minimizing the cost function $J = c^T\zeta$. Both the controller optimization and the plant optimization were performed using the interior point function **centers3** documented in Appendix C.

The first plant optimization code (**plantopt2**), which employs a single $\mathcal{H}_\infty$ constraint, provided the baseline structure for all that followed. Consequently, in the interest of volume, this code alone is thoroughly documented.

1. Plant and Controller Optimization for an $\mathcal{H}_\infty$ Performance Constraint

The function **plantopt2** and its associated subroutines solves the plant/ controller optimization problem with a pure $\mathcal{H}_\infty$ performance constraint. From Chapter V, the general method revolves about two EVP's. Algorithmically:

1. Evaluate $A(\zeta), B_1(\zeta)$, and $B_2(\zeta)$.
2. Determine $\mathcal{H}_\infty$ feasibility for the initial, and find a particular solution (W_p, Y_p) to the Riccati $\mathcal{H}_\infty$ analysis equation.
3. Create a basis Z for the controller variables W, and Y.
4. Find the vector ξ which maps Z to (W_p, Y_p).
5. iter Using $A(\zeta), B_1(\zeta)$, and $B_2(\zeta)$, and the basis Z, find the basis for R_1, the $\mathcal{H}_\infty$ LMI, equation 5.2.

6. Find a feasible initial eigenvalue λ_0 for the inequality $\lambda_0 I - R_1(\xi) > 0$.

7. Use the method of centers to find a controller (represented by a new ξ) which satisfies the $\mathcal{H}_\infty$ LMI:

   Minimize: λ (over ξ),

   Subject to:

$$\begin{bmatrix} \lambda I - R_1(\zeta, \xi) & 0 \\ 0 & Y(\xi) \end{bmatrix} > 0. \tag{D.1}$$

   Note that λ must be negative for the problem to be feasible.

8. Verify that λ represents a feasible solution, or that λ is not so small as to satisfy the termination criteria.

9. Evaluate $W(\xi), Y(\xi)$.

10. Using W and Y and the plant bases matrices $Aa, B1a$ and $B2a$, find a basis for the alternate $\mathcal{H}_\infty$ LMI (equation 5.3).

11. Now optimize the plant cost $J = c^T\zeta$, subject to the $\mathcal{H}_\infty$ constraint, and the constraint that the plant variables must have positive value:

    Minimize: J (over ζ),

    Subject to:

$$\begin{bmatrix} J - c^T\zeta & 0 & 0 \\ 0 & \mathrm{diag}(\zeta) & 0 \\ 0 & 0 & -R_2(\zeta, \xi) \end{bmatrix} > 0. \tag{D.2}$$

12. Evaluate the new plant $A(\zeta), B_1(\zeta)$, and $B_2(\zeta)$.

13. Go to step iter until exit criteria is satisfied.

The progress of this algorithm is supported by a number of subroutines. Feasibility is tested by Riccati solvers from the μ-Tools toolbox to determine an initial feasible solution. Note that this particular solution is not precisely feasible in

the since of the LMI's, since it represents the solution of a Riccati equality rather than inequality. The inequality will be enforced the first pass through the algorithm, and the Riccati solution being "close" to a solution of the inequality dramatically reduced the computational workload on the first pass. From the plant and a-matrices for the controller variables, the basis matrices for the constraint functional on ξ are constructed by the subroutine `hinfres6`. The interior point algorithm,`centers3`, then finds a controller that minimizes the controller margin. The controller margin must be negative for the controller to be considered feasible and for the algorithm to proceed. Furthermore, if the absolute value of the controller margin is less than some small numerical threshold (typically 10^{-6} to 10^{-10}), then the feasible set is too small practically to proceed, and the loop is exited. This is the usual exit path. If these criterion are satisfied, the code proceeds to the plant optimization phase. Using the above controller, the function `hinfres4` builds the basis matrices for the constraint functional on ζ. The interior point code is then called again to optimize the plant cost (over ζ). The loop can also be exited at this point if the cost does not decrease from iteration to iteration, or if the maximum iteration count has been reached. If the cost has been successfully decreased, then the algorithm returns to the controller design phase to find a new controller. The optimal plant, controller, and algorithm history are returned as outputs. The other called subroutines are administrative, and are described in Appendix C.

plantopt2

```
function [zeta,xi,K,Z,Rc,Tc,J2,err]=...
                              plantopt2(Aa,B1a,B2a,C,D,zeta,weight,thres,gam,theta,prec)
%
% [zeta,xi,K,Z,Rc,Tc,J2,err]=plantopt2(Aa,B1a,B2a,C,D,zeta,weight,thres,gam,theta,prec)
% [zeta,xi,K,Z,Rc,Tc,J2,err]=plantopt2(Aa,B1a,B2a,C,D,zeta,weight,thres)
```

```
%
% Description:
% Given:
% xdot =  A x + B1 w + B2 u,
%  z   =  C x +       D  u,
%  y   =    x,
%
% This function determines the minimum state-feedback plant satisfying
% ||T_zw||_infty < gamma
%
% min J=weight*zeta
% subject to:
%
% R(Z)=AY+YA'+B2W +W'B2'+B1B1'+(CY +DW)' (CY +DW)< 0,
%
%  Code alternatively finds central controller, and minimizing plant
%  using the method of centers to solve equivalent LMI's.
%
% Inputs:
%
% C,D are constant matrices from the state-space realization of the system.
% Aa, B1a and B2a are a-matrices which hold the affine elements
% zeta is the vector that defines the initial plant (A=affin(AA,zeta))
% weight is the row vector of weights in the objective function
% thres is the threshold on the central controller margin
%               for determining when to quit the iterative procedure
% gam is a bound of the H-inf norm for the closed loop system.
% theta parameter of the method of centers
% prec is used by the method of centers to determine when to quit the newton
%     search for the analytic center. The default is 0.001, bur if the ricatti
%     soln indicates that the problem is feasible and yet the method of centers
%     cannot find a solution, then this parameter should be adjusted to something
%     smaller to allow the newton search to go deeper.
%     If the ricatti solvers indicate that the problem is feasible and yet the
%     method of centers cannot find a feasible controller, then this parameter
%     should be set smaller.
%
% Outputs:
%
% z is the optimized vector of plant parameters
% xi is the vector of controller parameters
% K the state-feedback gain matrix
% Z is the basis for the controller matrices W,Y
% Rc, Tc are the bases for the controller and plant optimization constraints
% J2 is the history of the problem
% err is an error flag
%

% Called functions:
% basis, computes a basis for a set of block structured matrices
% hinfres, computes the a-matrix of a Riccati inequality restriction
```

```
% adiag, computes the a-matrix of diag(A(x),B(x))
% atrace, computes the a-matrix of trace(R,X)
% aident, gives an a-matrix with the indept. term equals unity and
%         the rest of matrices zeros.
% getvec, gives the realization of a matrix in a matrix basis.
% affin, computes an affin matrix function.
%
% Comments:
%
% The assumptions for this function are standard in the state-feedback
% Hinf case, (A,B1) stabilizable and D1 has full column rank.
% These assumptions are not checked, though they probably should be.
% If the inputs gam, prec and theta are not provided, they are initialized
% to 1, 0.1 and 0.001 respectively.

if nargin<11
   gam=1;        theta=0.1;            prec=.001;
   if nargin==7,
      thres=1e-6
   end
end

[ma,na]=size(Aa);                  dimz=na/ma;
[mb1,nb1]=size(B1a);               nq1=nb1/dimz;
[mb2,nb2]=size(B2a);               nq2=nb2/dimz;
[mc,nc]=size(C);

% Determine initial plant
A=affin(Aa,zeta);      B1=affin(B1a,zeta,nq1);    B2=affin(B2a,zeta,nq2);

%  Determine hinf feasibility/central controller
%   by solving hinf synthesis hamiltonian
a=A-B2*((D'*D)\D')*C;
ham=[a, ((B1*B1'/gam^2)-B2*((D'*D)\B2')); -C'*(eye(mc)-D*((D'*D)\D'))*C, -a'];
[x1,x2,fail]=ric_schr(ham);          Xinf=x2/x1;
if (fail>0) | any(eig(Xinf)<0),
   disp('Initial System appears infeasible-1');   return
else
   disp('Initial Hinf problem feasible')
end
Kp=-(D'*D)\(D'*C+B2'*Xinf);   % "central" controller

%  Determine particular soln from central controller
%   by solving hinf analysis hamiltonian
a1=A+B2*Kp;    b1=[B1,10000*sqrt(eps)*eye(ma)];     c1=C+D*Kp;
[x1,x2,fail]=ric_schr([a1',c1'*c1;-b1*b1',-a1]);        Yp=x2/x1;
if (fail>0) | any(eig(Yp)<0),
   disp('Initial System appears infeasible-2');   return
end
Wp=Kp*Yp;
```

```
%
% Initializing the optimization
%
[Z,YY,dimz]=basis2(ma,nq2);          % builds the basis Z for W,Y
xi=getvec2([Yp;Wp],Z);               % get initial xi from initial controller
Syy=[1,ma];                          % establishes the structure of Y
Rb=eye(ma+mc,(dimz+1)*(ma+mc));      % est. basis for controler EVP
Rc=[zeros(ma,ma),YY];
Ta=[0, weight];                      % est. basis for plant EVP
Tb=eye(1,length(Ta));
J=weight*zeta;
lam=-1;
J2=[];   iteration=0;
CD=[C D];                            % simplifies argument

while lam<-1e-12,

   iteration=iteration+1

   %
   % For fixed plant- find the optimal controller (best controller margin)
   %

   disp('searching for new controller')
   % create the basis for the hinf constraint
   [Ra,Sc2]=hinfres6([A B2],CD,B1*B1',gam,Z);

   % determine initial lambda for EVP and add some slop to make it feasible
   % this takes care of changing the equality above to inequality
   lamin=max(eig(affin(Ra,xi))) + 0.1;

   % find the set of controller parameters 'xi' which minimize the
   % controller margin 'lam'
   [xiop,lam,stat,err2]=centers3(Ra,Rb,Rc,Syy,xi,lamin,theta,prec,1e12);

   if err2~=0,
      err=3;      return;
   % terminate if centers couldn't find a feasible controller (lam>=0),
   % or the margin was too small to bother proceeding (lam>-thres)
   elseif lam>=0,                    % feasibility criteria
      disp('centers could not find a new feasible controller')
      break
   elseif lam>-thres,                % termination criteria
      disp('feasible set too small to proceed')
      break
   end

   % build new controller W,Y from xi
   YW=affin([zeros(ma+nq2,ma),Z],xiop,ma);
   Y=YW(1:ma,:);   W=YW(ma+1:ma+nq2,:);
   xi=xiop;
```

```
   K=W/Y;

   %
   % Fix controller- solve the min airframe problem
   %
   disp('Optimizing plant for previous controller')
   % using the controller above, find the basis for hinf restriction
   [Tc,Sc]=hinfres4(Aa,B1a,B2a,YW,CD,gam);

   % add some padding to previous optimum cost to make problem very feasible
   J=J+.1;

   % optimize the plant
   [zetaop,J,stat,err2]=centers3(Ta,Tb,Tc,Sc,zeta,J,theta,prec,1e12);
   if err2~=0
      err=3;
      return;
   end
   zeta=zetaop;

   % Build the new plant
   A=affin(Aa,zeta);   B1=affin(B1a,zeta,nq1);   B2=affin(B2a,zeta,nq2);
   disp([lam,J])

   % Stack the data for output history
   J2=[J2; lam, J, zeta'];
   disp('Closed Loop Poles=')
   E=eig(A+B2*K);      disp(E')
   if any(real(E)>=0),
      err=4;
      return
   end
   % alternate termination criteria (this is only a back-up to above)
   if iteration>2,
      if (J==J2(iteration-1,2)) | iteration==100,
         break
      end
   end

end

% Determine the controller

Y=YW(1:ma,:);
W=YW(ma+1:ma+nq2,:);
K=W/Y;

% plot the optimization history
loglog(-J2(:,1),J2(:,2),'*'),grid,title('Plant Cost vs. Controller Margin')
xlabel('Controller Margin'),ylabel('Plant Cost')
```

```
err=0;
return;

% end plantopt2
```

hinfres6

```
function [F,Sf]=hinfres6(AB,CD,B1B1,gam,Z)
%
% [F,Sf]=hinfres6(AB,CD,B1B1,gam,Z)
%
% Description:
%
% This function determines the a-matrix of the convex restriction
%
%           | AB*YW + (AB*YW)' + B1B1   (CD*YW)' |
% R(Z) =    |                             2      | < 0,
%           |      CD*YW               -gam I    |
%
%
% Inputs:
%
% AB,CD,B1B1,gam parameters of the restriction
% Z basis for YW from basis2
%
% Outputs:
%
% F a-matrix of the restriction
% Sf block structure of F
%
% Comments:
%
% Note that in order to obtain the value of the restriction, we need a
% vector x=[x1,...,xr], then F(x)=F0+F1*x1+...Fr*xr, the block structure
% of F(x) is Sf.
%
% this basis uses the reduced order problem

[mz,nz]=size(Z);
[mm,nm]=size(AB);
[mr,nr]=size(CD);
[ms,ns]=size(B1B1);
dim=nz/mm;                          mmr=mm+mr;
F=zeros(mmr,mmr*dim);
F(:,1:mmr)=[B1B1,zeros(mm,mr);zeros(mr,mm),-gam*gam*eye(mr)];
F3=zeros(mr);
for i=1:dim,
   YW=Z(:,(i-1)*mm+1:(i-1)*mm+mm);
   F1=AB*YW;                        F2=CD*YW;
```

```
   F(:,i*mmr+1:(i+1)*mmr)=[F1+F1',F2';F2,F3];
end
Sf=[1,mm+mr];
return;

% end hinfres6
```

hinfres4

```
function [R,Sf]=hinfres4(Aa,B1a,B2a,YW,CD,gam)
%
% [R,Sf]=hinfres4(Aa,B1a,B2a,YW,CD,gam)
%
% Description:
%
% This function determines the a-matrix of the convex restriction
%           -                                        -
%           | AY+B2W+YA'+W'B2'+(CY+DW)'(CY+DW)    B1  |
% R(zeta) = |                                         | < 0,
%           |      B1'                          - eye |
%           -                                        -
%
% Inputs:
%
% Aa,B1a,B2a are a-matrices of the plant
% YW,CD,gam parameters of the restriction (YW is the packed form [Y;W])
%
% Outputs:
%
% R a-matrix of the restriction (-R>0 is actually returned)
% Sf block structure of R
%
% Comments:
%
% Note that in order to obtain the value of the restriction, we need a
% vector x=[x1,...,xr], then R(x)=R0+R1*x1+...Rr*xr, the block structure
% of R(x) is Sf.
%
[ma,na]=size(Aa);                          dim=na/ma;
[mb1,nb1]=size(B1a);                       nq1=nb1/dim;
[mb2,nb2]=size(B2a);                       nq2=nb2/dim;
dimz=dim-1;                                i1=dimz+ma+nq1;
F1=[Aa(:,1:ma),B2a(:,1:nq2)]*YW;
CYDW=CD*YW/gam;                     CYDW2= CYDW'*CYDW;
R=zeros(i1,dim*(i1));                      F3=zeros(nq1);
R(dim:i1,dim:i1)=-[F1+F1'+CYDW2, B1a(:,1:nq1); B1a(:,1:nq1)',-eye(nq1)];
for i=1:dimz,
   R(i,i+i1*i)=1;
```

```
   F1=[Aa(:,i*ma+1:i*ma+ma),B2a(:,i*nq2+1:i*nq2+nq2)]*YW;
   F2=B1a(:,i*nq1+1:i*nq1+nq1);
   R(dim:i1,(i*i1+dim):(i*i1+i1))=-[F1+F1',F2;F2',F3];
end
Sf=[zeros(dimz,1),ones(dimz,1);1,ma+nq1];
return;

% end hinfres4
```

2. Plant and Controller Optimization for an $\mathcal{H}_\infty$ Performance Constraint at Multiple Flight Conditions

The function **plantopt5** solved the multiple flight condition problem. It called the identical subroutines as plantopt2 above, and differed only slightly in structure. During the first phase three independent controllers were calculated for the three respective flight conditions. The second phase then stacked the constraint functionals so as to jointly perform the plant optimization. The subroutine adiag performed the administrative task of stacking a-matrices for the imposition of joint constraints, and is documented in Appendix C.

plantopt5

```
function [zeta,xi1,xi2,xi3,Z,J2,err]=...
plantopt5(Aa,B1a,B2a,Aa2,B1a2,B2a2,Aa3,B1a3,B2a3,C,D,zeta,weight,gam,theta,prec)
%
% [K,val,err]=h2hinfsf(Aa,B1a,B2a,C,D,zeta,weight,gam,theta,prec)
% [K,val,err]=h2hinfsf(Aa,B1a,B2a,C,D,zeta,weight,gam)
%
% Description:
% Given:
% xdot =  A x + B1 w + B2 u,
%  z   =  C x +       D  u,
%  y   =    x,
%
% This function determines the minimum statefeedback plant satisfying
% ||T_zw||_infty < gamma
% min J=weight'*zeta
% subject to
%
```

```
% R(Z)=AY+YA'+B2W +W'B2'+B1B1'+(CY +DW)' (CY +DW)< 0,
%
%  Code alternatively finds controller, and minimizing plant
%  using the method of centers.
%
%  THIS CODE IS THE ANALOG TO PLANTOPT2 AND
%  FINDS THE SIMULTANEOUS OPTIMUM FOR
%  MULTIPLE FLIGHT CONDITIONS
%
% Inputs:
%
% C,D are constant matrices from the state-space realization of the system.
% Aa, B1a and B2a are a-matrices which hold the affine elements
% zeta is the vector that defines the initial plant (A=affin(AA,zeta))
% weight is the row vector of weights in the objective function
% gam is a bound of the H-inf norm for the closed loop system.
% theta parameter of the method of centers
% prec precision in the compute of the upper bound of the H2 norm.
%
% Outputs:
%
% K the gain
% val the optimum value of the performance index computed by centers
%     algorithm
% err error code, its value is err=1 if the problem is infeasible,
%     otherwise err=0.

% Called functions:
% basis, computes a basis for a set of block structured matrices
% hinfres, computes the a-matrix of a Riccati inequality restriction
% adiag, computes the a-matrix of diag(A(x),B(x))
% atrace, computes the a-matrix of trace(R,X)
% aident, gives an a-matrix with the indept. term equals unity and
%         the rest of matrices zeros.
% getvec, gives the realization of a matrix in a matrix basis.
% affin, computes an affin matrix function.
%
% Comments:
%
% The assumptions for this function are standard in the state-feedback
% Hinf case, (A,B1) stabilizable and D1 has full column rank.
% These assumptions are not checked, though they probably should be.
% If the inputs prec and theta are not provided, they are initialized
% to 0.1 and 0.001 respectively.
if nargin==8
   theta=0.1;            prec=.001;
end
[ma,na]=size(Aa);                  dimz=na/ma;
[mb1,nb1]=size(B1a);               nq1=nb1/dimz;
[mb2,nb2]=size(B2a);               nq2=nb2/dimz;
[mc,nc]=size(C);
```

```
%  Flight Condition 1
%  Determine feasibility/controller
A=affin(Aa,zeta);      B1=affin(B1a,zeta,nq1);    B2=affin(B2a,zeta,nq2);
a=A-B2*((D'*D)\D')*C;
ham=[a, ((B1*B1'/gam^2)-B2*((D'*D)\B2')); -C'*(eye(mc)-D*((D'*D)\D'))*C, -a'];
[x1,x2,fail]=ric_schr(ham);          Xinf=x2/x1;
if (fail>0) | any(eig(Xinf)<0),
   disp('Initial System appears infeasible-1');   return
else
   disp('Initial Hinf problem feasible')
end
Kp=-(D'*D)\(D'*C+B2'*Xinf);   % controller

%  Determine particular soln from controller
a1=A+B2*Kp;    b1=[B1,10000*sqrt(eps)*eye(ma)];     c1=C+D*Kp;
[x1,x2,fail]=ric_schr([a1',c1'*c1;-b1*b1',-a1]);        Yp1=x2/x1;
if (fail>0) | any(eig(Yp1)<0),
   disp('Initial System appears infeasible-2');   return
end
Wp1=Kp*Yp1;

%  Flight Condition 2
%  Determine feasibility/controller
A2=affin(Aa2,zeta);      B12=affin(B1a2,zeta,nq1);    B22=affin(B2a2,zeta,nq2);
a=A2-B22*((D'*D)\D')*C;
ham=[a, ((B12*B12'/gam^2)-B22*((D'*D)\B22'));-C'*(eye(mc)-D*((D'*D)\D'))*C,-a'];
[x1,x2,fail]=ric_schr(ham);          Xinf=x2/x1;
if (fail>0) | any(eig(Xinf)<0),
   disp('Initial System appears infeasible-1');   return
else
   disp('Initial Hinf problem feasible')
end
Kp=-(D'*D)\(D'*C+B22'*Xinf);   % controller

%  Determine particular soln from controller
a1=A2+B22*Kp;    b1=[B12,10000*sqrt(eps)*eye(ma)];     c1=C+D*Kp;
[x1,x2,fail]=ric_schr([a1',c1'*c1;-b1*b1',-a1]);        Yp2=x2/x1;
if (fail>0) | any(eig(Yp2)<0),
   disp('Initial System appears infeasible-2');   return
end
Wp2=Kp*Yp2;

%  Flight Condition 3
%  Determine feasibility/controller
A3=affin(Aa3,zeta);      B13=affin(B1a3,zeta,nq1);    B23=affin(B2a3,zeta,nq2);
a=A3-B23*((D'*D)\D')*C;
ham=[a, ((B13*B13'/gam^2)-B23*((D'*D)\B23'));-C'*(eye(mc)-D*((D'*D)\D'))*C,-a'];
[x1,x2,fail]=ric_schr(ham);          Xinf=x2/x1;
if (fail>0) | any(eig(Xinf)<0),
   disp('Initial System appears infeasible-1');   return
```

```
else
   disp('Initial Hinf problem feasible')
end
Kp=-(D'*D)\(D'*C+B2'*Xinf);   % controller

%  Determine particular soln from controller
a1=A3+B23*Kp;    b1=[B13,10000*sqrt(eps)*eye(ma)];     c1=C+D*Kp;
[x1,x2,fail]=ric_schr([a1',c1'*c1;-b1*b1',-a1]);        Yp3=x2/x1;
if (fail>0) | any(eig(Yp3)<0),
   disp('Initial System appears infeasible-2');   return
end
Wp3=Kp*Yp3;

%
% Initializing the optimization
%
[Z,YY,dimz]=basis2(ma,nq2);          Syy=[1,ma];
Rb=eye(ma+mc,(dimz+1)*(ma+mc));      Rc=[zeros(ma,ma),YY];
CD=[C D];
Ta=[0, weight];                      Tb=eye(1,length(Ta));
J=weight*zeta;                       lam=-1;
xi1=getvec2([Yp1;Wp1],Z);            J2=[];  iteration=0;
xi2=getvec2([Yp2;Wp2],Z);
xi3=getvec2([Yp3;Wp3],Z);

while lam<-1e-12,

   iteration=iteration+1
   %
   % defining/solving the controller problem-flt cond 1
   %
   disp('searching for new controller- flt cond 1')
   [Ra,Sc2]=hinfres6([A B2],CD,B1*B1',gam,Z);
   lamin=max(eig(affin(Ra,xi1))) + 0.1;
   [xiop,lam1,stat,err2]=centers3b(Ra,Rb,Rc,Syy,xi1,lamin,theta,prec,1e12);
   if err2~=0,
      err=3;      return;
   elseif lam>=0,
      disp('centers could not find a new feasible controller')
      break
   elseif lam>-1e-6,
      disp('feasible set too small to proceed')
      break
   end
   YW1=affin([zeros(ma+nq2,ma),Z],xiop,ma);
   xi1=xiop;

   %
   % defining/solving the controller problem-flt cond 2
   %
   disp('searching for new controller- flt cond 2')
```

```
[Ra,Sc2]=hinfres6([A2 B22],CD,B12*B12',gam,Z);
lamin=max(eig(affin(Ra,xi2))) + 0.1;
[xiop,lam2,stat,err2]=centers3b(Ra,Rb,Rc,Syy,xi2,lamin,theta,prec,1e12);
if err2~=0,
   err=3;      return;
elseif lam2>=0,
   disp('centers could not find a new feasible controller (fc2)')
   break
end
YW2=affin([zeros(ma+nq2,ma),Z],xiop,ma);
xi2=xiop;

%
% defining/solving the controller problem-flt cond 3
%
disp('searching for new controller- flt cond 3')
[Ra,Sc2]=hinfres6([A3 B23],CD,B13*B13',gam,Z);
lamin=max(eig(affin(Ra,xi3))) + 0.1;
[xiop,lam3,stat,err2]=centers3b(Ra,Rb,Rc,Syy,xi3,lamin,theta,prec,1e12);
if err2~=0,
   err=3;      return;
elseif lam3>=0,
   disp('centers could not find a new feasible controller (fc3)')
   break
end
YW3=affin([zeros(ma+nq2,ma),Z],xiop,ma);
xi3=xiop;

%
% defining/solving the min airframe problem
%
disp('Optimizing plant for previous controller')
[Tc1,Sc1]=hinfres4(Aa,B1a,B2a,YW1,CD,gam);
[Tc2,Sc2]=hinfres4(Aa2,B1a2,B2a2,YW2,CD,gam);
[Tc3,Sc3]=hinfres4(Aa3,B1a3,B2a3,YW3,CD,gam);
% use 'adiag' to stack joint constraints
Tc=adiag(Tc1,Tc2);          Tc=adiag(Tc,Tc3);
Sc=[Sc1;Sc2;Sc3];
J=J+1000*eps;
[zetaop,J,stat,err2]=centers3b(Ta,Tb,Tc,Sc,zeta,J,theta,prec,1e12);
if err2~=0
   err=3;
   return;
end
zeta=zetaop;
A=affin(Aa,zeta);     B1=affin(B1a,zeta,nq1);     B2=affin(B2a,zeta,nq2);
A2=affin(Aa2,zeta);   B12=affin(B1a2,zeta,nq1);   B22=affin(B2a2,zeta,nq2);
A3=affin(Aa3,zeta);   B13=affin(B1a3,zeta,nq1);   B23=affin(B2a3,zeta,nq2);

J1=[lam1,lam2,lam3,J];
disp(J1)
```

```
   J2=[J2; lam1,lam2,lam3,J];
   if iteration>2,
      if (J>0.999999*J2(iteration-1,4)) | iteration==50,
         break
      end
   end

end

% Determine the controller

Y1=YW1(1:ma,:);    W1=YW1(ma+1:ma+nq2,:);    K1=W1/Y1;
Y2=YW2(1:ma,:);    W2=YW2(ma+1:ma+nq2,:);    K2=W2/Y2;
Y3=YW3(1:ma,:);    W3=YW3(ma+1:ma+nq2,:);    K3=W3/Y3;

loglog(-J2(:,1),J2(:,4),'*',-J2(:,2),J2(:,4),'+',-J2(:,3),J2(:,4),'x'),
grid,title('Plant Cost vs. Controller Margin')
xlabel('Controller Margin'),ylabel('Plant Cost')

err=0;
return;

% end plantopt5.m
```

3. Joint $\mathcal{H}_\infty$ Pole Placement Plant/Controller Optimization

The function **plantopt6** solves the plant/controller optimization problem where joint $\mathcal{H}_\infty$ and pole placement performance constraints are imposed. The structure is identical to the optimization functions above, and the algorithm follows Section D.. The administrative functions **hinfres6** and **hinfres4** above create the basis matrices for the $\mathcal{H}_\infty$ constraint, while **poleres1** and **poleres2** create the basis matrices for the pole placement constraint. The administrative function **adiag** is then used to create the joint bases.

The significant difference between this code and the previous codes is that the Riccati solvers prior to the beginning of the iterative routines could only find a controller that satisfies the $\mathcal{H}_\infty$ constraint, which was no guarantee of the existence of a joint controller. The first pass through the algorithm therefore started

with a controller that was $\mathcal{H}_\infty$ feasible, but which most likely failed to satisfy the pole placement constraint. The first pass therefore determined the feasibility of the problem. As mentioned in Chapter V, the failure of the algorithms to find a jointly feasible state-feedback controller did not by necessity mean that a feasible controller did not exist.

plantopt6

```
function
[zeta,xi,K,Z,Ra,Tc,J2,err]=...
              plantopt6(Aa,B1a,B2a,C,D,zeta,weight,rad,alpha,theta,prec)
%
% [K,val,err]=h2hinfsf(Aa,B1a,B2a,C,D,zeta,weight,gam,theta,prec)
% [K,val,err]=h2hinfsf(Aa,B1a,B2a,C,D,zeta,weight,gam)
%
% Description:
% Given:
% xdot =  A x + B1 w + B2 u,
%  z   =  C x +        D  u,
%  y   =    x,
%
% This function determines the minimum statefeedback plant satisfying
% ||T_zw||_infty < 1, and closed-loop poles in the circular disc of radius r
% centered at (-(rad + alpha),0),
%
% min J=weight'*zeta
% subject to the following
%
% R(Z)=AY+YA'+B2W +W'B2'+B1B1'+(CY +DW)' (CY +DW)< 0,
%
% and
%       -                                                          -
%       | A1Y+YA1'+B2W(I+A1/r)'+(I+A1/r)W'B2'+ A1YA1'/r    B2*W    |
% R(Z)= |                                                          | < 0,
%       |      W'B2'                                       -Y/r    |
%       -                                                          -
%
%   where A1=A + alpha*eye(A)
%
%  Code alternatively finds controller, and minimizing plant
%  using the method of centers.
%
% Inputs:
%
% C,D are constant matrices from the state-space realization of the system.
% Aa, B1a and B2a are a-matrices which hold the affine elements
```

```
% zeta is the vector that defines the initial plant (A=affin(AA,zeta))
% weight is the row vector of weights in the objective function
% rad and alpha specify the circle
% theta parameter of the method of centers
% prec precision in the compute of the upper bound of the H2 norm.
%
% Outputs:
%
% K the gain
% val the optimum value of the performance index computed by centers
%     algorithm
% err error code, its value is err=1 if the problem is infeasible,
%     otherwise err=0.

% Called functions:
% basis, computes a basis for a set of block structured matrices
% hinfres, computes the a-matrix of a Riccati inequality restriction
% adiag, computes the a-matrix of diag(A(x),B(x))
% atrace, computes the a-matrix of trace(R,X)
% aident, gives an a-matrix with the indept. term equals unity and
%         the rest of matrices zeros.
% getvec, gives the realization of a matrix in a matrix basis.
% affin, computes an affin matrix function.
%
% Comments:
%
% The assumptions for this function are standard in the state-feedback
% Hinf case, (A,B2) stabilizable and D1 has full column rank.
% These assumptions are not checked, though they probably should be.
% If the inputs theta and prec are not provided, they are initialized
% to 0.1 and 0.001 respectively. If the circle parameters are not specified
% then they are set for a circle of radius 87.3, centered at -97.3 on the real
% axis.
if nargin<=9,
   theta=0.1;            prec=.001;
   if nargin<9,
      rad=87.3;                          alpha=0.5;
   end
end
[ma,na]=size(Aa);                  dimz=na/ma;
[mb1,nb1]=size(B1a);               nq1=nb1/dimz;
[mb2,nb2]=size(B2a);               nq2=nb2/dimz;
[mc,nc]=size(C);

%  Determine hinf feasibility/controller
gam=1;
A=affin(Aa,zeta);      B1=affin(B1a,zeta,nq1);    B2=affin(B2a,zeta,nq2);
a=A-B2*((D'*D)\D')*C;
ham=[a, ((B1*B1'/gam^2)-B2*((D'*D)\B2')); -C'*(eye(mc)-D*((D'*D)\D'))*C, -a'];
[x1,x2,fail]=ric_schr(ham);          Xinf=x2/x1;
if (fail>0) | any(eig(Xinf)<0),
```

```
   disp('Initial System appears infeasible-1');   return
else
   disp('Initial Hinf problem feasible')
end
Kp=-(D'*D)\(D'*C+B2'*Xinf);   % controller

%  Determine particular soln from controller
a1=A+B2*Kp;    b1=[B1,10000*sqrt(eps)*eye(ma)];     c1=C+D*Kp;
[x1,x2,fail]=ric_schr([a1',c1'*c1;-b1*b1',-a1]);
Yp=x2/x1;
if (fail>0) | any(eig(Yp)<0),
   disp('Initial System appears infeasible-2');   return
end
Wp=Kp*Yp;

%
% Initializing the optimization
%
[Z,YY,dimz]=basis2(ma,nq2);          Syy=[1,ma];
Rb=eye(3*ma+mc,(dimz+1)*(3*ma+mc)); Rc=[zeros(ma,ma),YY];
CD=[C D];
Ta=[0, weight];                      Tb=eye(1,length(Ta));
J=weight*zeta;                       lam=-1;
xi=getvec2([Yp;Wp],Z);               J2=[];   iteration=0;

while lam<-1e-6,

   iteration=iteration+1
   %
   % defining/solving the controller problem
   %
   disp('searching for new controller')
   [Ra1,Sa1]=hinfres6([A B2],CD,B1*B1',gam,Z);
   [Ra2,Sa2]=poleres1(A,B2,rad,alpha,Z);
   % use adiag to form joint constraint
   Ra=adiag(Ra1,Ra2);           Sac=[Sa1;Sa2;Syy];
   lamin=max(eig(affin(Ra,xi))) + 0.1;
   [xiop,lam,stat,err2]=centers3d(Ra,Rb,Rc,Sac,xi,lamin,theta,prec,1e12);
   if err2~=0,
      err=3;      return;
   elseif lam>=0,
      disp('centers could not find a new feasible controller')
      break
   elseif lam>-1e-6,
      disp('feasible set too small to proceed')
      break
   end
   YW=affin([zeros(ma+nq2,ma),Z],xiop,ma);
   Y=YW(1:ma,:);   W=YW(ma+1:ma+nq2,:);
   xi=xiop;
   K=W/Y;
```

```
   %
   % defining/solving the min airframe problem
   %
   disp('Optimizing plant for previous controller')
   [Tc1,Sc1]=hinfres4(Aa,B1a,B2a,YW,CD,gam);
   [Tc2,Sc2]=poleres2(Aa,B2a,rad,alpha,Y,W);
   Tc=adiag(Tc1,Tc2);          Stac=[Sc1;Sc2];
   J=J+0.1;
   [zetaop,J,stat,err2]=centers3d(Ta,Tb,Tc,Stac,zeta,J,theta,prec,1e12);
   if err2~=0
      err=3;
      return;
   end
   zeta=zetaop;
   A=affin(Aa,zeta);   B1=affin(B1a,zeta,nq1);   B2=affin(B2a,zeta,nq2);
   disp('[Lambda, J]=')
   disp([lam,J])
   disp('Closed Loop Poles=')
   E=eig(A+B2*K);      disp(E')
   if any(real(E)>=0),
      err=4;
      return
   end
   J2=[J2; lam, J,zeta'];
   if iteration>2,
      if (J==J2(iteration-1,2)) | iteration==100,
         break
      end
   end

end

disp('Final Closed Loop Poles=')
E=eig(A+B2*K);
disp(E)

loglog(-J2(:,1),J2(:,2),'*'),grid,
xlabel('Controller Margin'),ylabel('Plant Cost')

err=0;
return;

% end plantopt6.m
```

poleres1

```
function [F,Sf]=poleres1(A,B,rad,alpha,Z)
```

```
%
% [F,Sf]=poleres1(A,B,rad,alpha,Z)
%
% Description: Pole Placement constraint.
%
% This function determines the a-matrix of the convex restriction
%       -                                                       -
%       | A1Y+YA1'+BW(I+A1/r)'+(I+A1/r)W'B'+ A1YA1'/r    B*W    |
% R(Z)= |                                                       | < 0,
%       |      W'B'                                      -Yr    |
%       -                                                       -
%
% Inputs:
%
% A,B,rad,alpha parameters of the restriction
% Z basis for YW from basis2
%
% Outputs:
%
% F a-matrix of the restriction
% Sf block structure of F
%
% Comments:
%
% Note that in order to obtain the value of the restriction, we need a
% vector x=[x1,...,xr], then F(x)=F0+F1*x1+...Fr*xr, the block structure
% of F(x) is Sf.
%
% this basis is for pole placement

[mz,nz]=size(Z);
[ma,na]=size(A);
[mb,nb]=size(B);
dim=nz/ma;                      mm=2*ma;
F=zeros(mm,mm*dim);
A1=A+alpha*eye(ma);
for i=1:dim,
   Y=Z(1:ma,(i-1)*ma+1:(i-1)*ma+ma);
   W=Z(ma+1:ma+nb,(i-1)*ma+1:(i-1)*ma+ma);
   F2=B*W;
   F1=A1*Y + (A1/rad + eye(ma))*F2';
   F(:,i*mm+1:(i+1)*mm)=[F1+F1'+ (A1*Y*A1'/rad),F2;F2',-Y*rad];
end
Sf=[1,mm];
return;

% end poleres1
```

poleres2

```
function [F,Sf]=poleres2(Aa,B2a,rad,alpha,Y,W)
%
% [F,Sf]=poleres2(Aa,B2a,rad,alpha,Y,W)
%
% Description:
%
% This function determines the a-matrix of the convex restriction
%        -                                                                -
%        | (A+B*K+alpha*eye)*Y+Y*(A+B*K+alpha*eye)'  (A+B*K+alpha*eye) |
% R(Z)= -|                                                                | > 0,
%        |      (A+B*K+alpha*eye)                       -inv(Y/r)       |
%        -                                                               -
%
% Inputs:
% rad,alpha,Y,W parameters of the restriction
% Aa, B2a bases for the plant
%
% Outputs:
% F a-matrix of the restriction   (returns R>0)
% Sf block structure of F
%
% Comments:
%
% Note that in order to obtain the value of the restriction, we need a
% vector x=[x1,...,xr], then F(x)=F0+F1*x1+...Fr*xr, the block structure
% of F(x) is Sf.
%
% this basis is for pole placement

[ma,na]=size(Aa);              mm=2*ma;
[mb,nb]=size(B2a);
dim=na/ma;                     nq2=nb/dim;
F=zeros(mm,mm*dim);            Fz=zeros(ma,ma);
K=W/Y;

Ai=Aa(:,1:ma);
Bi=B2a(:,1:nq2);
F1=Ai+Bi*K+eye(ma);
F(:,1:mm)=-[F1*Y+Y*F1',F1;F1',-inv(Y/rad)];

for i=2:dim,
   Ai=Aa(:,(i-1)*ma+1:i*ma);
   Bi=B2a(:,(i-1)*nq2+1:i*nq2);
   F1=Ai+Bi*K;
   F(:,(i-1)*mm+1:i*mm)=-[F1*Y+Y*F1',F1;F1',Fz];
end
Sf=[1,mm];
return;
% end poleres2
```

4. Plant/Controller Optimization with a Joint $\mathcal{H}_\infty$ Static Maneuverability Specification

The design code `plantopt2a` is a minor modification to the baseline $\mathcal{H}_\infty$ plant optimization code `plantopt2`. The static maneuverability specification must be posed as the basis (in ζ) to an LMI external to the routine. This basis `Da` is then passed to `plantopt2a` as an input variable. During the plant optimization phase of the routine, `Da` is diagonally augmented to the basis for the $\mathcal{H}_\infty$ specification using `adiag`. Because the typical termination conditions are different than the standard plant optimization problem, the termination criteria was modified slightly. The code that follows is otherwise identical to `plantopt2`, and required no additional subroutines.

plantopt2a

```
function [zeta,xi,K,Z,Rc,Tc,J2,err]=...
            plantopt2a(Aa,B1a,B2a,C,D,Da,zeta,weight,thres,gam,theta,prec)
%
%[zeta,xi,K,Z,Rc,Tc,J2,err]=...
%            plantopt2a(Aa,B1a,B2a,C,D,Da,zeta,weight,thres,gam,theta,prec)
%
% Description:
% Given:
% xdot =  A x + B1 w + B2 u,
%  z   =  C x +       D  u,
%  y   =    x,
%
% This function determines the minimum state-feedback plant satisfying
% ||T_zw||_infty < gamma and a static manuverability spec posed by Da.
%
% min J=weight'*zeta
% subject to
%
%  R(xi,zeta)=   AY+YA'+B2W +W'B2'+ B1B1'+(CY +DW)'(CY +DW) < 0,
%                                                      -
%  and
```

```
%
%  D(zeta) > 0
%
%  Code alternatively finds central controller, and minimizing plant
%  using the method of centers.
%
% Inputs:
%
% C,D are constant matrices from the state-space realization of the system.
% Aa, B1a and B2a are a-matrices which hold the affine elements
% zeta is the vector that defines the initial plant (A=affin(AA,zeta))
% weight is the row vector of weights in the objective function
% thres is the threshold on the central controller margin
%               for determining when to quit the iterative procedure
% gam is a bound of the H-inf norm for the closed loop system.
% theta parameter of the method of centers
% prec is used by the method of centers to determine when to quit the newton
%     search for the analytic center. The default is 0.001, bur if the ricatti
%     soln indicates that the problem is feasible and yet the method of centers
%     cannot find a solution, then this parameter should be adjusted to something
%     smaller to allow the newton search to go deeper.
%     If the ricatti solvers indicate that the problem is feasible and yet the
%     method of centers cannot find a feasible controller, then this parameter
%     should be set smaller.
%
% Outputs:
%
% K the gain
% val the optimum value of the performance index computed by centers
%     algorithm
% err error code, its value is err=1 if the problem is infeasible,
%     otherwise err=0.

% Called functions:
% basis, computes a basis for a set of block structured matrices
% hinfres, computes the a-matrix of a Riccati inequality restriction
% adiag, computes the a-matrix of diag(A(x),B(x))
% atrace, computes the a-matrix of trace(R,X)
% aident, gives an a-matrix with the indept. term equals unity and
%         the rest of matrices zeros.
% getvec, gives the realization of a matrix in a matrix basis.
% affin, computes an affin matrix function.
%
% Comments:
%
% The assumptions for this function are standard in the state-feedback
% Hinf case, (A,B1) stabilizable and D1 has full column rank.
% These assumptions are not checked, though they probably should be.
% If the inputs gam, prec and theta are not provided, they are initialized
% to 1, 0.1 and 0.001 respectively.
```

```
if nargin<11
   gam=1;        theta=0.1;            prec=.001;
   if nargin==7,
      thres=1e-6
   end
end

[ma,na]=size(Aa);                  dimz=na/ma;
[mb1,nb1]=size(B1a);               nq1=nb1/dimz;
[mb2,nb2]=size(B2a);               nq2=nb2/dimz;
[mc,nc]=size(C);

%  Determine hinf feasibility/central controller
A=affin(Aa,zeta);      B1=affin(B1a,zeta,nq1);    B2=affin(B2a,zeta,nq2);
a=A-B2*((D'*D)\D')*C;
ham=[a, ((B1*B1'/gam^2)-B2*((D'*D)\B2')); -C'*(eye(mc)-D*((D'*D)\D'))*C, -a'];
[x1,x2,fail]=ric_schr(ham);          Xinf=x2/x1;
if (fail>0) | any(eig(Xinf)<0),
   disp('Initial System appears infeasible-1');   return
else
   disp('Initial Hinf problem feasible')
end
Kp=-(D'*D)\(D'*C+B2'*Xinf);   % "central" controller

%  Determine particular soln from central controller
a1=A+B2*Kp;    b1=[B1,10000*sqrt(eps)*eye(ma)];     c1=C+D*Kp;
[x1,x2,fail]=ric_schr([a1',c1'*c1;-b1*b1',-a1]);        Yp=x2/x1;
if (fail>0) | any(eig(Yp)<0),
   disp('Initial System appears infeasible-2');   return
end
Wp=Kp*Yp;

%
% Initializing the optimization
%
[Z,YY,dimz]=basis2(ma,nq2);          Syy=[1,ma];
Rb=eye(ma+mc,(dimz+1)*(ma+mc));      Rc=[zeros(ma,ma),YY];
CD=[C D];
Ta=[0, weight];                      Tb=eye(1,length(Ta));
J=weight*zeta;                       lam=-1;
xi=getvec2([Yp;Wp],Z);               J2=[];   iteration=0;

while lam<-1e-12,

   iteration=iteration+1
   %
   % defining/solving the central controller problem
   %
   disp('searching for new central controller')
   [Ra,Sc2]=hinfres6([A B2],CD,B1*B1',gam,Z);
   lamin=max(eig(affin(Ra,xi))) + 0.1;
```

```
   [xiop,lam,stat,err2]=centers3(Ra,Rb,Rc,Syy,xi,lamin,theta,prec,1e12);
   if err2~=0,
      err=3;      return;
   elseif lam>=0,
      disp('centers could not find a new feasible controller')
      break
   elseif lam>-thres,
      disp('feasible set too small to proceed')
      break
   end
   YW=affin([zeros(ma+nq2,ma),Z],xiop,ma);
   Y=YW(1:ma,:);   W=YW(ma+1:ma+nq2,:);
   xi=xiop;
   K=W/Y;

   %
   % defining/solving the min airframe problem
   %
   disp('Optimizing plant for previous controller')
   [Tc1,Sc1]=hinfres4(Aa,B1a,B2a,YW,CD,gam);
   Tc = adiag(Tc1,Da);
   Sc = [Sc1;1 1];
   J=J+.1;
   [zetaop,J,stat,err2]=centers3(Ta,Tb,Tc,Sc,zeta,J,theta,prec,1e12);
   if err2~=0
      err=3;
      return;
   end
   zeta=zetaop;
   A=affin(Aa,zeta);   B1=affin(B1a,zeta,nq1);   B2=affin(B2a,zeta,nq2);
   disp([lam,J])
   J2=[J2; lam, J, zeta'];
   disp('Closed Loop Poles=')
   E=eig(A+B2*K);      disp(E')
   if any(real(E)>=0),
      err=4;
      return
   end
   if iteration>2,
      if (J>0.999999*J2(iteration-1,2)) | iteration==100,
         break
      end
   end

end

% Determine the controller

Y=YW(1:ma,:);
W=YW(ma+1:ma+nq2,:);
K=W/Y;
```

```
loglog(-J2(:,1),J2(:,2),'*'),grid,title('Plant Cost vs. Controller Margin')
xlabel('Central Controller Margin'),ylabel('Plant Cost')

err=0;
return;

% end plantopt2a.m
```

5. Plant/Controller Optimization with Dynamic Maneuverability Constraints

As discussed in Chapter V, dynamic maneuverability requirements can either be imposed using a closed-loop $\mathcal{H}_\infty$ constraint or second formulation based upon determinantal relationships. For the first method, the original $\mathcal{H}_\infty$ code `plantopt2` is suitable without modification. The following code, `plantopt9` use the Lyapunov formulation. The subroutines `actres1` and `actres2` perform the formation of the bases matrices of the LMI associated with this constraint.

In practice, `plantopt2` was used for Example problem 5, and the following code routinely resulted in the method of centers departing the feasible set. This suggests a feature in the geometry of the constraint which `centers3` could not handle. Given more reliable interior point codes, and the fact that both formulations were conservative, it would be interesting to see which of the two formulations for the dynamic maneuverability requirements resulted in a lower final plant cost.

plantopt9

```
function [zeta,xi,K,Z,Ra,Tc,J2,err]=...
          plantopt9(Aa,B1a,B2a,B3a,C,D,zeta,weight,umax,thres,theta,prec)
%
% [zeta,xi,K,Z,Ra,Tc,J2,err]=...
%            plantopt9(Aa,B1a,B2a,B3a,C,D,zeta,weight,umax,theta,prec)
% [zeta,xi,K,Z,Ra,Tc,J2,err]=plantopt9(Aa,B1a,B2a,B3a,C,D,zeta,weight,umax)
%
% Description:
```

```
% Given:
% xdot =  A x + B1 w + B2 u,
%  z   =  C x +       D  u,
%  y   =    x,
%
% This function determines the minimum state-feedback plant satisfying
% ||T_zw||_infty < 1, and  open-loop actuator limitation,
%
% min J=weight'*zeta
% subject to the following
%
%   R(Z)= AY+YA'+B2W +W'B2' + B1B1' +(CY+DW)(CY +DW)' < 0
%
% and
%
%   F(Z)=AY+B2W+B3W(i,:)/u_max(i) + (AY+B2W+B3W(i,:)/u_max(i))' < 0
%
%  Code alternatively finds central controller, and minimizing plant
%  using the method of centers.
%
% Inputs:
%
% C,D are constant matrices from the state-space realization of the system.
% Aa, B1a, B3a and B2a are a-matrices which hold the affine elements
%       B1a corresponds to the exogenous disturbance inputs
%       B3a corresponds to the exogenous command inputs
% zeta is the vector that defines the initial plant (A=affin(AA,zeta))
% weight is the row vector of weights in the objective function
% thres determines the central controller margin threshold at which
%       the set of feasible control is considered too small to proceed
%       Should usually be set to 1e-6 to 1e-10. Default=1e-9
% prec  establishes how deep the newton search will proceed looking for
%       the analytic center. Default=1e-6
%
% Outputs:
% z2    output vector zeta of minimized plant parameters
% xi    vector of optimal controller parameters
%

% Called functions:
% basis, computes a basis for a set of block structured matrices
% hinfres, computes the a-matrix of a Riccati inequality restriction
% adiag, computes the a-matrix of diag(A(x),B(x))
% atrace, computes the a-matrix of trace(R,X)
% aident, gives an a-matrix with the indept. term equals unity and
%         the rest of matrices zeros.
% getvec, gives the realization of a matrix in a matrix basis.
% affin, computes an affin matrix function.
%
% Comments:
%
```

```
% The assumptions for this function are standard in the state-feedback
% Hinf case, (A,B2) stabilizable and D1 has full column rank.
% These assumptions are not checked, though they probably should be.
% If the inputs theta and prec are not provided, they are initialized
% to 0.1 and 0.001 respectively.
if nargin==9,
  thres=1e-9;       theta=0.1;            prec=1e-6;
else nargin<9,
   error('Insufficent number of input arguments')
end
[ma,na]=size(Aa);                  dimz=na/ma;
[mb1,nb1]=size(B1a);               nq1=nb1/dimz;
[mb2,nb2]=size(B2a);               nq2=nb2/dimz;
[mc,nc]=size(C);

%  Determine hinf feasibility/central controller
gam=1;
A=affin(Aa,zeta);      B1=affin(B1a,zeta,nq1);    B2=affin(B2a,zeta,nq2);
B3=affin(B3a,zeta,1);
a=A-B2*((D'*D)\D')*C;
ham=[a, ((B1*B1'/gam^2)-B2*((D'*D)\B2')); -C'*(eye(mc)-D*((D'*D)\D'))*C, -a'];
[x1,x2,fail]=ric_schr(ham);          Xinf=x2/x1;
if (fail>0) | any(eig(Xinf)<0),
   disp('Initial System appears infeasible-1');   return
else
   disp('Initial Hinf problem feasible')
end
Kp=-(D'*D)\(D'*C+B2'*Xinf);   % central controller

%  Determine particular soln from central controller
a1=A+B2*Kp;    b1=[B1,10000*sqrt(eps)*eye(ma)];     c1=C+D*Kp;
[x1,x2,fail]=ric_schr([a1',c1'*c1;-b1*b1',-a1]);
Yp=x2/x1;
if (fail>0) | any(eig(Yp)<0),
   disp('Initial System appears infeasible-2');   return
end
Wp=Kp*Yp;

%
% Initializing the optimization
%
[Z,YY,dimz]=basis2(ma,nq2);          Syy=[1,ma];
Rb=eye(3*ma+mc,(dimz+1)*(3*ma+mc));  Rc=[zeros(ma,ma),YY];
CD=[C D];
Ta=[0, weight];                      Tb=eye(1,length(Ta));
J=weight*zeta;                       lam=-1;
xi=getvec2([Yp;Wp],Z);               J2=[];   iteration=0;

while lam<-1e-12,

   iteration=iteration+1
```

```
%
% defining/solving the central controller problem
%
disp('searching for new central controller')
[Ra1,Sa1]=hinfres6([A B2],CD,B1*B1',gam,Z);
[Ra2,Sa2]=actres1(A,B3,B2,umax,Z);
Ra=adiag(Ra1,Ra2);          Sac=[Sa1;Sa2;Syy];
lamin=max(eig(affin(Ra,xi))) + 0.1;
[xiop,lam,stat,err2]=centers3d(Ra,Rb,Rc,Sac,xi,lamin,theta,prec,1e12);
if err2~=0,
   err=3;      return;
elseif lam>=0,
   disp('centers could not find a new feasible controller')
   break
elseif lam>-thres,
   disp('feasible set too small to proceed')
   break
end
YW=affin([zeros(ma+nq2,ma),Z],xiop,ma);
Y=YW(1:ma,:);   W=YW(ma+1:ma+nq2,:);
xi=xiop;
K=W/Y;

%
% defining/solving the min airframe problem
%
disp('Optimizing plant for previous controller')
[Tc1,Sc1]=hinfres4(Aa,B1a,B2a,YW,CD,gam);
[Tc2,Sc2]=actres2(Aa,B3a,B2a,umax,W,Y);
Tc=adiag(Tc1,Tc2);         Stac=[Sc1;Sc2];
J=J+10;
[zetaop,J,stat,err2]=centers3d(Ta,Tb,Tc,Stac,zeta,J,theta,prec,1e12);
if err2~=0
   err=3;
   return;
end
zeta=zetaop;
A=affin(Aa,zeta);   B1=affin(B1a,zeta,nq1);   B2=affin(B2a,zeta,nq2);
B3=affin(B3a,zeta,1);
disp('[Lambda, J]=')
disp([lam,J])
disp('Closed Loop Poles=')
E=eig(A+B2*K);      disp(E')
if any(real(E)>=0),
   err=4;
   return
end
J2=[J2; lam, J,zeta'];
if iteration>2,
   if (J==J2(iteration-1,2)) | iteration==100,
      break
```

```
      end
   end

end

disp('Final Closed Loop Poles=')
E=eig(A+B2*K);
disp(E)

loglog(-J2(:,1),J2(:,2),'*'),grid,
xlabel('Central Controller Margin'),ylabel('Plant Cost')

err=0;
return;

% end plantopt6.m
```

actres1

```
function [F,Sf]=actres1(A,B3,B2,umax,Z)
%
% [F,Sf]=actres1(A,B1,B2,umax,Z)
%
% Description: Open-Loop constraint.
%
% This function determines the a-matrix of the affine restriction
%
%  F(Z)=AY+B2W+B3W(i,:)/u_max(i) + (AY+B2W+B3W(i,:)/u_max(i))' < 0
%
% Inputs:
%
% A,B1,B2,umax parameters of the restriction
% Z basis for YW from basis2
%
% Outputs:
%
% F a-matrix of the restriction
% Sf block structure of F
%
% Comments:
%
% Note that in order to obtain the value of the restriction, we need a
% vector x=[x1,...,xr], then F(x)=F0+F1*x1+...Fr*xr, the block structure
% of F(x) is Sf.
%
% this basis is for open loop actuator limitations
%

nu=length(umax);
[mz,nz]=size(Z);
```

```
[ma,na]=size(A);
[mb3,nb3]=size(B3);
[mb2,nb2]=size(B2);
dim=nz/ma;
F=[]; Sf=[];
for k=1:nu,
   F0=zeros(ma,ma*(1+dim));
   for i=1:dim,
      Y=Z(1:ma,(i-1)*ma+1:(i-1)*ma+ma);
      W=Z(ma+1:ma+nb2,(i-1)*ma+1:(i-1)*ma+ma);
      F1=A*Y+B2*W+B3/umax(k)*W(k,:);
      F0(:,i*ma+1:(i+1)*ma)=F1+F1';
   end
   F=adiag(F,F0);
   Sf=[Sf;1,ma];
end
return;

% end actres1
```

actres2

```
function [F,Sf]=actres2(Aa,B3a,B2a,umax,W,Y)

% [F,Sf]=actres2(Aa,B3a,B2a,umax,W,Y)
%
% Description: Open-Loop constraint.
%
% This function determines the a-matrix of the affine restriction
%
% F(Z)=-(AY+B2W+B3W(i,:)/u_max(i) + (AY+B2W+B3W(i,:)/u_max(i))') > 0
%
% Inputs:
%
% umax,Y,W parameters of the restriction
% Aa,B1a,B2a bases for the plant parameters
%
% Outputs:
%
% F a-matrix of the restriction
% Sf block structure of F
%
% Comments:
%
% Note that in order to obtain the value of the restriction, we need a
% vector x=[x1,...,xr], then F(x)=F0+F1*x1+...Fr*xr, the block structure
% of F(x) is Sf.
%
```

```
% this basis is for open loop actuator limitations
%

nu=length(umax);
[ma,na]=size(Aa);
[mb3,nb3]=size(B3a);
[mw,nw]=size(W);
dim=na/ma;
if nb3>dim, error('B3 matrix needs to have one column'),end
F=[]; Sf=[];
for k=1:nu,
   F0=zeros(ma,ma*dim);
   for i=1:dim,
      Ai=Aa(:,(i-1)*ma+1:i*ma);
      B3i=B3a(:,i);
      B2i=B2a(:,(i-1)*mw+1:i*mw);
      F1=Ai*Y+B2i*W+B3i/umax(k)*W(k,:);
      F0(:,(i-1)*ma+1:i*ma)=-(F1+F1');
   end
   F=adiag(F,F0);
   Sf=[Sf;1,ma];
end
return;

% end actres2
```

6. Plant/Controller Optimization with Robustness Constraints

As discussed in Chapter V, the $\mathcal{H}_\infty$ constraint can be used to pose either disturbance rejection or robustness constraints on a plant optimization problem. The formulation of robustness constraints is slightly different than the imposition of turbulence rejection specifications due to the structure of the problem. A separate plant optimization function (`plantopt10`) was consequently required, as well as an additional subroutine (`hinfres9`) which could form the basis for the robustness constraint.

plantopt10

```
function [zeta,xi,K,Z,Ra,Tc,J2,err]=...
              plantopt10(Aa,B1a,B2a,C,D,B3,C3a,D3a,zeta,weight,thres,theta,prec)
```

```
%
%   [zeta,xi,K,Z,Ra,Tc,J2,err]=...
%               plantopt10(Aa,B1a,B2a,C,D,B3,C3a,D3a,zeta,weight,thres,theta,prec)
%
%
% Description:
% Given:
% xdot =  A(zeta) x + B1(zeta) w + B2(zeta) u + B3 w3,
%  z   =  C       x +                   D  u,
%  z3  =  C3(zeta)x +              D3(zeta) u
%  y   =          x,
%
% This function determines the minimum statefeedback plant satisfying
% ||T_zw||_infty < 1, and ||T_z3w3||_infty < 1:
%
% min J=weight'*zeta
% subject to the following
%
% R(Z)= AY+YA'+B2W +W'B2'+B1B1'+(CY +DW)'(CY +DW)  < 0,
%
% and
%
% R(Z)= AY+YA'+B2W +W'B2'+B3B3'+(C3Y +D3W)'(C3Y +D3W)  < 0,
%
%  Code alternatively finds central controller, and minimizing plant
%  using the method of centers.
%  This function was specifically intended to solve the hinf/hinf
%  problem where T_z3w3 is the uncertainty transfer function
%  (and the plant parameters show up in (C3,D3)).
%
% Inputs:
%
% C,D,B3 are constant matrices from the state-space realization of the system.
% Aa, B1a, B2a,C3a, and D3a are a-matrices which hold the affine elements
% zeta is the vector that defines the initial plant (A=affin(AA,zeta))
% weight is the row vector of weights in the objective function
% thres determines the controller margin below which the routine quits
% theta parameter of the method of centers
% prec precision in the compute of the upper bound of the H2 norm.
%
% Outputs:
%
% K the gain
% err error code, its value is err=1 if the problem is infeasible,
%     otherwise err=0.

% Called functions:
% basis, computes a basis for a set of block structured matrices
% hinfres, computes the a-matrix of a Riccati inequality restriction
% adiag, computes the a-matrix of diag(A(x),B(x))
% atrace, computes the a-matrix of trace(R,X)
```

```
% aident, gives an a-matrix with the indept. term equals unity and
%         the rest of matrices zeros.
% getvec, gives the realization of a matrix in a matrix basis.
% affin, computes an affin matrix function.
%
% Comments:
%
% The assumptions for this function are standard in the state-feedback
% Hinf case, (A,B2) stabilizable and D1 has full column rank.
% These assumptions are not checked, though they probably should be.

[ma,na]=size(Aa);                    dimz=na/ma;
[mb1,nb1]=size(B1a);                 nq1=nb1/dimz;
[mb2,nb2]=size(B2a);                 nq2=nb2/dimz;
[mc,nc]=size(C);
[mc3,nc3]=size(C3a);

A=affin(Aa,zeta);       B1=affin(B1a,zeta,nq1);    B2=affin(B2a,zeta,nq2);
C3=affin(C3a,zeta,ma);  D3=affin(D3a,zeta,nq2);


%  Determine hinf feasibility/central controller
gam=1;
a=A-B2*((D'*D)\D')*C;
ham=[a, ((B1*B1'/gam^2)-B2*((D'*D)\B2')); -C'*(eye(mc)-D*((D'*D)\D'))*C, -a'];
[x1,x2,fail]=ric_schr(ham);          Xinf=x2/x1;
if (fail>0) | any(eig(Xinf)<0),
   disp('Initial System appears infeasible-1');   return
else
   disp('Initial Hinf problem feasible')
end
Kp=-(D'*D)\(D'*C+B2'*Xinf);   % central controller

%  Determine particular soln from central controller
a1=A+B2*Kp;    b1=[B1,10000*sqrt(eps)*eye(ma)];     c1=C+D*Kp;
[x1,x2,fail]=ric_schr([a1',c1'*c1;-b1*b1',-a1]);
Yp=x2/x1;
if (fail>0) | any(eig(Yp)<=0),
   disp('Initial System appears infeasible-2');   return
end
Wp=Kp*Yp;

%
% Initializing the optimization
%
[Z,YY,dimz]=basis2(ma,nq2);          Syy=[1,ma];
Rb=eye(2*ma+mc+mc3,(dimz+1)*(2*ma+mc+mc3));
Rc=[zeros(ma,ma),YY];
CD=[C D];
Ta=[0, weight];                      Tb=eye(1,length(Ta));
J=weight*zeta;                       lam=-1;
```

```
xi=getvec2([Yp;Wp],Z);              J2=[];   iteration=0;

while lam<-1e-12,

   iteration=iteration+1
   %
   % defining/solving the controller problem
   %
   disp('searching for new controller')
   [Ra1,Sa1]=hinfres6([A B2],CD,B1*B1',gam,Z);
   [Ra2,Sa2]=hinfres6([A B2],[C3 D3],B3*B3',gam,Z);
   Sac=[Sa1;Sa2;Syy];
   Ra=adiag(Ra1,Ra2);
   lamin=max(eig(affin(Ra,xi)));
   if lamin>=0,
      lamin=lamin*1.1;
   else,
      lamin=lamin+0.1;
   end
   [xiop,lam,stat,err2]=centers3d(Ra,Rb,Rc,Sac,xi,lamin,theta,prec,1e24);
   if err2~=0,
      err=3;      return;
   elseif lam>=0,
      disp('centers could not find a new feasible controller')
      break
   elseif lam>-thres,
      disp('feasible set too small to proceed')
      break
   end
   YW=affin([zeros(ma+nq2,ma),Z],xiop,ma);
   Y=YW(1:ma,:);   W=YW(ma+1:ma+nq2,:);
   xi=xiop;
   K=W/Y;

   %
   % defining/solving the min airframe problem
   %
   disp('Optimizing plant for previous controller')
   [Tc1,Sc1]=hinfres4(Aa,B1a,B2a,YW,CD,gam);
   [Tc2,Sc2]=hinfres9(Aa,B2a,B3,C3a,D3a,YW)
   Tc=adiag(Tc1,Tc2);          Stac=[Sc1;Sc2];
   J=J+1;
   [zetaop,J,stat,err2]=centers3d(Ta,Tb,Tc,Stac,zeta,J,theta,prec,1e24);
   if err2~=0
      err=3;
      return;
   end
   zeta=zetaop;
   A=affin(Aa,zeta);    B1=affin(B1a,zeta,nq1);   B2=affin(B2a,zeta,nq2);
   C3=affin(C3a,zeta,ma);  D3=affin(D3a,zeta,nq2);
   disp('[Lambda, J]=')
```

```
   disp([lam,J])
   disp('Closed Loop Poles=')
   E=eig(A+B2*K);      disp(E')
   if any(real(E)>=0),
      err=4;
      return
   end
   J2=[J2; lam, J,zeta'];
   if iteration>2,
      if (J==J2(iteration-1,2)) | iteration==100,
         break
      end
   end

end

disp('Final Closed Loop Poles=')
E=eig(A+B2*K);
disp(E)

if isempty(J2), return, end
loglog(-J2(:,1),J2(:,2),'*'),grid,
xlabel('Central Controller Margin'),ylabel('Plant Cost')

err=0;
return;

% end plantopt10.m
```

hinfres9

```
function [R,Sf]=hinfres9(Aa,B2a,B3,C3a,D3a,YW)
%
% [R,Sf]=hinfres4(Aa,B1a,B2a,YW,CD,gam)
%
% Description:
%
% This function determines the a-matrix of the convex restriction
%
%           -                                          -
%           | AY+B2W+YA'+W'B2'+B3B3'        (C3Y+D3W)'  |
% R(zeta) = |                                          | < 0,
%           |           (C3Y+D3W)               - eye  |
%           -                                          -
% This subroutine was written for use with plantopt10
% Note that zeta_i > 0 is not enforced since that is done by hinfres4
%
% Inputs:
%
```

```
% Aa,B2a,C3a,D3a are a-matrices of the plant
% YW,B3 constant parameters of the restriction (YW is the packed form [Y;W])
%
% Outputs:
%
% R a-matrix of the restriction (-R>0 is actually returned)
% Sf block structure of R
%
% Comments:
%
% Note that in order to obtain the value of the restriction, we need a
% vector x=[x1,...,xr], then R(x)=R0+R1*x1+...Rr*xr, the block structure
% of R(x) is Sf.
%

[ma,na]=size(Aa);                              dim=na/ma;
[mb2,nb2]=size(B2a);                           nq2=nb2/dim;
[mc3,nb3]=size(C3a);
i1=ma+mc3;
R=zeros(i1,dim*(i1));
F1=[Aa(:,1:ma),B2a(:,1:nq2)]*YW;
F2=[C3a(:,1:ma),D3a(:,1:nq2)]
R(1:i1,1:i1)=-[F1+F1'+B3*B3', F2';F2 ,-eye(nq1)];
F3=zeros(mc3);
for i=1:dim-1,
   F1=[Aa(:,i*ma+1:i*ma+ma),B2a(:,i*nq2+1:i*nq2+nq2)]*YW;
   F2=[C3a(:,i*ma+1:i*ma+ma),D3a(:,i*nq2+1:i*nq2+nq2)]*YW;
   R(dim:i1,(i*i1+dim):(i*i1+i1))=-[F1+F1',F2';F2,F3];
end
Sf=[1,il];
return;

% end hinfres9
```

7. Joint $\mathcal{H}_2$ / Pole-Placement Plant/Controller Optimization

A design code was also produced to solve the plant/controller optimization problem subject to a joint $\mathcal{H}_2$ /pole-placement performance constraint (`plantopt8b`). The $\mathcal{H}_2$ performance measure was not considered to be as useful as the $\mathcal{H}_\infty$ problem, and it was not used for any of the example problems. This code is consequently included for archival purposes only. The subroutines `h2res1` and `h2res2` prepared the basis matrices for the $\mathcal{H}_2$ constraint. Interested readers should consult [Ref. 29] for a discussion of the generalized $\mathcal{H}_2$ cost. The derivation of the expressions used in

these subroutines is straightforward from the expressions in this reference.

plantopt8b

```
function
[zeta,xi,K,Z,Ra,Rc,J2,err]=...
          plantopt8(Aa,B1a,B2a,C,D,zeta,weight,gam,theta,prec)
%
% [zeta,xi,K,Z,Ra,Tc,J2,err]=plantopt8(Aa,B1a,B2a,C,D1,D2,zeta,weight,gam,theta,prec)
% [zeta,xi,K,Z,Ra,Tc,J2,err]=plantopt8(Aa,B1a,B2a,C,D1,D2,zeta,weight)
%
% Description:
% Given:
% xdot =  A x + B1 w + B2 u,
%  z   =  C x +      + D u,
%  y   =    x,
%
% This function determines the minimum plant and associated state-feedback
% controller satisfying ||T_zw||_2(gen'l) < 1, and closed-loop poles in the
% circular disc of radius r,  centered at (-(rad + alpha),0). The
%
% min J=weight'*zeta
% subject to the following
%           -
%           | AY+YA'+B2W +W'B2'     B1          0               0   |
% R1(Z)=    |       B1'           -eye          0               0   | < 0,
%           |       0               0     CYC'+DWC'+CW'D'-eye   DW  |
%           |       0               0          W'D'              Y  |
%           -
% and
%             -
%             | A1Y+YA1'+B2W(I+A1/r)'+(I+A1/r)W'B2'+ A1YA1'/r    B2*W   |
% R2(Z)=      |                                                         | < 0,
%             |      W'B2'                                       -Y*r   |
%
%
%   where A1=A + alpha*eye(A)
%
%  Code alternatively finds controller, and minimizing plant
%  using the method of centers.
%
% Inputs:
%
% C,D1,D2 are constant matrices from the state-space realization of the system.
% Aa, B1a and B2a are a-matrices which hold the affine elements
% zeta is the vector that defines the initial plant (A=affin(AA,zeta))
% weight is the row vector of weights in the objective function
% gam is a bound of the H-inf norm for the closed loop system.
% theta parameter of the method of centers
```

```
% prec precision in the compute of the upper bound of the H2 norm.
%
% Outputs:
%
% K the gain
% val the optimum value of the performance index computed by centers
%     algorithm
% err error code, its value is err=1 if the problem is infeasible,
%     otherwise err=0.

% Called functions:
% basis2, computes a basis for Y and W
% h2res,  computes the a-matrix of the H-2 restriction
% poleres, computes the a-matrix of the pole placement restriciton
% adiag, computes the a-matrix of diag(A(x),B(x))
% atrace, computes the a-matrix of trace(R,X)
% aident, gives an a-matrix with the indept. term equals unity and
%         the rest of matrices zeros.
% getvec2, gives the realization of a matrix in a matrix basis.
% affin, computes an affin matrix function.
%
% Comments:
%
% The assumptions for this function are standard in the state-feedback
% Hinf case, (C1,A,B2) detectable and stabilizable and D2 has full column rank.
% These assumptions are not checked, though they probably should be.

% If the inputs gam,theta and prec are not provided, they are initialized
% to 1,0.1 and 0.001 respectively.
if nargin==8,
   gam=1;      theta=0.1;            prec=.001;
end
[ma,na]=size(Aa);                  dimz=na/ma;
[mb1,nb1]=size(B1a);               nq1=nb1/dimz;
[mb2,nb2]=size(B2a);               nq2=nb2/dimz;
[mc,nc]=size(C);

% Inital Plant
A=affin(Aa,zeta);      B1=affin(B1a,zeta,nq1);    B2=affin(B2a,zeta,nq2);

% Find a good starting point (in the circle,Yp>0, and (A+BK)Y+Y(A+BK)'+B1B1'<0)
Poles=[-5:-1:-4-ma];
Kp=place(A,B2,Poles);                  % warning: place returns A-BK stable
Yp=lyap((A-B2*Kp),B1*B1');
Yp=(Yp+Yp')/2;                         % this corrects for inaccuracies in lyap
Wp=-Kp*Yp;
[Z,YY,dimz]=basis2(ma,nq2);
xi=getvec2([Yp;Wp],Z);

% Admin
CD=[C D];
```

```
Ta=[0, weight];                    Tb=eye(1,length(Ta));
J=weight*zeta;                     lam=-1;
J2=[];   iteration=0;

% Set the circle criterion
rad=87.3;                          alpha=1;

while lam<-1e-6,

   iteration=iteration+1
   %
   % defining/solving the controller problem
   %
   disp('Searching for new controller')
   [R1,S1,R2,S2]=h2res1(3,[A B2],C,D,B1*B1',Z);
   [R3,S3]=poleres1(A,B2,rad,alpha,Z);  % poleres implicitly enforces Y>0
   if iteration==1,
      % Mathematically the initial xi found above should satisfy all but R2
      % Numerically it doesn't work that well so we first need to find a feasible xi
      Ra=adiag(R1,R2);                 Ra=adiag(Ra,R3);
      Rb=eye(size(Ra));
      Rc=[zeros(ma,ma),YY];
      Sac=[S1;S2;S3;1,ma];
   else,
      Ra=R1;
      % Note that the controller margin jumps since the objective function changes
      Rb=eye(ma,(dimz+1)*ma);
      Rc=adiag(-R2,-R3);
      Sac=[S1;S2;S3];
   end
   lamin=max(eig(affin(Ra,xi))) + 0.1;
   [xiop,lam,stat,err2]=centers3d(Ra,Rb,Rc,Sac,xi,lamin,theta,prec,1e12);
   if err2~=0,
      err=3;      return;
   elseif lam>=0,
      disp('centers could not find a new feasible controller')
      break
   elseif (iteration>1) & (lam>-1e-6),
      disp('feasible set too small to proceed')
      break
   end
   YW=affin([zeros(ma+nq2,ma),Z],xiop,ma);
   Y=YW(1:ma,:);   W=YW(ma+1:ma+nq2,:);
   xi=xiop;
   K=W/Y;

   %
   % defining/solving the min airframe problem
   %
   disp('Optimizing plant for previous controller')
   [Tc1,Sc1]=h2res2(Aa,B1a,B2a,YW);
```

```
   [Tc2,Sc2]=poleres2(Aa,B2a,rad,alpha,Y,W);
   Tc=adiag(Tc1,Tc2);          Stac=[Sc1;Sc2];
   J=J+0.1;
   [zetaop,J,stat,err2]=centers3d(Ta,Tb,Tc,Stac,zeta,J,theta,prec,1e12);
   if err2~=0
      err=3;
      return;
   end
   zeta=zetaop;
   A=affin(Aa,zeta);   B1=affin(B1a,zeta,nq1);   B2=affin(B2a,zeta,nq2);
   disp('[Margin, J]=')
   disp([lam,J])
   disp('Closed Loop Poles=')
   E=eig(A+B2*K);      disp(E')
   if any(real(E)>=0),
      err=4;
      return
   end
   J2=[J2; lam, J];
   if iteration>2,
      if (J==J2(iteration-1,2)) | iteration==100,
         break
      end
   end

end

disp('Final Closed Loop Poles=')
E=eig(A+B2*K);
disp(E)

loglog(-J2(:,1),J2(:,2),'*'),grid,
xlabel('Controller Margin'),ylabel('Plant Cost')

err=0;
return;

% end plantopt8.m
```

h2res1

```
function [F,Sf,R,Sr]=h2res1(f,AB,C,D,B1B1,Z)
%
% [F,Sf,R,Sr]=h2res1(f,AB,C,D,B1B1,Z)
%
% Description:
%
% This function determines the a-matrices of the affine restrictions
```

```
% associated with the generalized H_2 constraint.
% The input 'f' specifies whether the H_2 constraint is to use the
% max eigenvalue (f=2) or the maximum diagonal element (f=3).
%
%       1) F=AB*YW + (AB*YW)' + B1B1   < 0,
%       and
%       2a) f=2:  R= |   CYC' +DWC'+CW'D'-eye     DW    |
%                    |   W'D'                    -Y     |   <0
%            or
%       2b) f=3:  R= | diag(diag(CYC' +DWC'+CW'D'-eye))    DW    |
%                    |   W'D'                              -Y    |   <0
%
%     Note that Y>0 is implicitly enforced within R<0.
%
% Inputs:
%
% AB,C,D,B1B1 parameters of the restriction
% Z basis for YW from basis2
%
% Outputs:
%
% F a-matrix of the restriction
% Sf block structure of F
%
% Comments:
%
% Note that in order to obtain the value of the restriction, we need a
% vector x=[x1,...,xr], then F(x)=F0+F1*x1+...Fr*xr, the block structure
% of F(x) is Sf.

[mz,nz]=size(Z);
[mc,nc]=size(C);
dim=nz/nc;                          mm=nc+mc;
F=zeros(nc,nc*dim);
R=zeros(mm,mm*dim);

F(:,1:nc)=B1B1;
R(1:mc,1:mc)=-eye(mc);

for i=1:dim,
   Y=Z(1:nc,(i-1)*nc+1:(i-1)*nc+nc);    W=Z(nc+1:mz,(i-1)*nc+1:(i-1)*nc+nc);
   F1=AB*[Y;W];
   DW=D*W;
   F(:,i*nc+1:(i+1)*nc)=F1+F1';
   if f==3,
      R(:,i*mm+1:(i+1)*mm)=[diag(diag(C*Y*C' +DW*C'+C*DW')),DW;DW',-Y];
   elseif f==2,
      R(:,i*mm+1:(i+1)*mm)=[(C*Y*C' +DW*C'+C*DW'),DW;DW',-Y];
   else,
      error('f must equal either 2 or 3 in h2res1')
   end
```

```
end
Sf=[1,nc];
Sr=[1,nc+mc];
return;

% end h2res1
```

h2res2

```
function [R,Sf]=h2res2(Aa,B1a,B2a,YW)
%
% [R,Sf]=h2res2(Aa,B1a,B2a,YW)
%
% Description:
%
% This function determines the a-matrix of the convex restriction
%              -                                      -
%           | AY+B2W+YA'+W'B2'           B1           |
% R(zeta) = |                                         | < 0,
%           |      B1'                       - eye    |
%              -                                      -
%   and zeta>0
%
% Note that the second part of the H_2 constraint (eye-(C+DK)Y(C+DK)'>0)
% is independent of zeta!
% Note also that R(zeta) is independent of the method by which the
% H-2 constraint is imposed.
%
% Inputs:
%
% Aa,B1a,B2a are a-matrices of the plant
% YW,gam parameters of the restriction (YW is the packed form [Y;W])
%
% Outputs:
%
% R a-matrix of the restriction (-R>0 is actually returned)
% Sf block structure of R
%
% Comments:
%
% Note that in order to obtain the value of the restriction, we need a
% vector x=[x1,...,xr], then R(x)=R0+R1*x1+...Rr*xr, the block structure
% of R(x) is Sf.
%
[ma,na]=size(Aa);                          dim=na/ma;
[mb1,nb1]=size(B1a);                       nq1=nb1/dim;
[mb2,nb2]=size(B2a);                       nq2=nb2/dim;
dimz=dim-1;                                i1=dimz+ma+nq1;
```

```
F1=[Aa(:,1:ma),B2a(:,1:nq2)]*YW;
R=zeros(i1,dim*(i1));                          F3=zeros(nq1);
R(dim:i1,dim:i1)=-[F1+F1', B1a(:,1:nq1); B1a(:,1:nq1)', -eye(nq1)];
for i=1:dimz,
   R(i,i+i1*i)=1;
   F1=[Aa(:,i*ma+1:i*ma+ma),B2a(:,i*nq2+1:i*nq2+nq2)]*YW;
   F2=B1a(:,i*nq1+1:i*nq1+nq1);
   R(dim:i1,(i*i1+dim):(i*i1+i1))=-[F1+F1',F2;F2',F3];
end
Sf=[zeros(dimz,1),ones(dimz,1);1,ma+nq1];
return;

% end h2res2
```

B. EXAMPLE SCRIPTS

This section documents the scripts which were used in the preparation of the example problems of Chapter V. The principal objective of these scripts is to take the linear aerodynamic coefficients and translate them into an open-loop state space representation that reflects the dependence on the optimization variables of interests. The outputs `Aa,B1a,B2a,C` and `D` are then the primary inputs to the optimization functions in the section above. The section below follow the order in which the examples were introduced in Chapter V.

1. Example 1- Optimal Vertical Tail at a Single Flight Condition

This script executes the directional dynamics example illustrated in Chapter V. The stability derivative data for a F-4 aircraft in the power approach flight condition was extracted from [Ref. 37]. The functional dependence of the tail size was determined using the relationships in [Ref. 40] to break the stability derivative data into its *tail* and *wing/body* contributions. The formulae for the plant can be found in equations 5.16. This script also provides the plant bases for Example problems 3 and 4.

f4 opt 1b

```
% Flight condition 1
% Approach

% Script establishes a directional aircraft plant for
% optimization of all-moving vertical tail.

% f-4 data
rho=0.00238;  % slugs/ft^3
V=230;        % fps
Izz=133700;   % slug-ft^2
S=530;        % ft^2          wing area
b=38.7;       % ft            wing span
Sb=243.86;    % ft^2          body surface area
lb=53.12;     % ft            body length
lt=19.8;      % ft            tail position from cg
Tmax = 20000;
le = 2.0;
umax = 30/57.3;

Cnbwb=-0.00107;     % yaw moment due to beta (wing/body)
Cybt=0.425;         % side force due to beta (tail)

q=0.5*rho*V*V;     % lbf/ft^2
sigmabeta=0.043;   % variance on disturbance

% build functional state-space representation
A0=[0 1;-Izz\q*S*b*Cnbwb -Izz\2*q*S*b*(lt/V)*Cnbwb];
A1=[0 0;-Izz\q*S*b*Cybt -Izz\2*q*S*b*(lt/V)*Cybt];
Aa=[A0 A1];

B10=[0 -Izz\q*S*b*Cnbwb]';
B11=[0 -Izz\q*S*b*Cybt]';
B1a=[B10 B11]*sigmabeta;

B20=[0 0]';
B21=[0 -Izz\q*S*b*Cybt]';
B2a=[B20 B21];

% initial parameters
Vto=0.47;             % original tail volume
Vro=0.3*Vto;          % original relative rudder volume
zeta=[Vto   ]';
weight=[1 ];

% original dynamics
Ao=affin(Aa,zeta);
B2o=affin(B2a,zeta,1);

% sigmabeta is variance on beta disturbances
```

```
B1=[0;-Izz\q*S*b*(Cnbwb+Cybt*Vto)]*sigmabeta;
C=[1/0.035 0; 0 0];
D=[0;1/0.175];

DRdyn=eig(Ao);
nat_frq=abs(DRdyn(1));
damping=-cos(angle(DRdyn(1)));

% Construct the basis for the static moment problem (example 4)
Do = -Tmax*le;
D1 = q*S*b*Cybt*umax;
Da = [Do,D1];

% end F4_opt_1b
```

2. Example 2- Optimal Vertical Tail at Multiple Flight Conditions

The following two scripts prepared the vertical tail optimization problem at high subsonic and supersonic flight conditions. These were then used in concert with the script in the above slow speed script to perform the multiple flight condition example problem. The synthesis model is different from the script above in two respects. First of all, the stability derivatives reflect the values appropriate to the various flight conditions. Secondly, the turbulence rejection specification is significantly different, and required different scaling of the input and output vectors w and z. Scaling of w was reflected in the C and D matrices, while scaling of w was reflected in the B_1 and D matrices.

f4 opt 2b.m

```
% Flight condition 2
% Subsonic cruise

% Script establishes a directional aircraft plant for vertical optimization.

% f-4 data
rho=0.000739; % slugs/ft^3
V=876;        % fps
Izz=139800;   % slug-ft^2
S=530;        % ft^2          wing area
```

```
b=38.7;       % ft           wing span
Sb=243.86;    % ft^2         body surface area
lb=53.12;     % ft body length
lt=19.8;      % ft           tail position from cg

Cnbwb=-0.00125;     % yaw moment due to beta (wing/body)
Cybt=0.266;         % side force due to beta (tail)

q=0.5*rho*V*V;     % lbf/ft^2
sigmabeta=0.0057;   % variance on disturbance

% build functional state-space representation
A0=[0 1;-Izz\q*S*b*Cnbwb -Izz\2*q*S*b*(lt/V)*Cnbwb];
A1=[0 0;-Izz\q*S*b*Cybt -Izz\2*q*S*b*(lt/V)*Cybt];
Aa2=[A0 A1];

B10=[0 -Izz\q*S*b*Cnbwb]';
B11=[0 -Izz\q*S*b*Cybt]';
B1a2=[B10 B11]*sigmabeta;

B20=[0 0]';
B21=[0 -Izz\q*S*b*Cybt]';
B2a2=[B20 B21];

% initial parameters
Vto=0.47;             % original tail volume
Vro=0.3*Vto;          % original relative rudder volume
zeta=[Vto ]';
weight=[1 ];

% sigmabeta is variance on beta disturbances
B1=[0;-Izz\q*S*b*(Cnbwb+Cybt*Vto)]*sigmabeta;
C=[1/0.035 0; 0 0];      % weights rms beta to be less than 2 deg
D=[0;1/0.175];           % weights rms rudder deflection to be less than 10 deg
```

f4 opt 3b.m

```
% Flight condition 3
% Supersonic cruise

% Script establishes a directional aircraft plant for  vertical tail optimization.

% f-4 data
rho=0.000287; % slugs/ft^3
V=1742;        % fps
Izz=139800;   % slug-ft^2
S=530;        % ft^2         wing area
b=38.7;       % ft           wing span
Sb=243.86;    % ft^2         body surface area
lb=53.12;       % ft body length
lt=19.8;          % ft           tail position from cg
```

```
Cnbwb=-0.00133;     % yaw moment due to beta (wing/body)
Cybt=0.193;         % side force due to beta (tail)

q=0.5*rho*V*V;     % lbf/ft^2
sigmabeta=0.0029;   % variance on disturbance

% build functional state-space representation
A0=[0 1;-Izz\q*S*b*Cnbwb -Izz\2*q*S*b*(lt/V)*Cnbwb];
A1=[0 0;-Izz\q*S*b*Cybt -Izz\2*q*S*b*(lt/V)*Cybt];
Aa3=[A0 A1 ];

B10=[0 -Izz\q*S*b*Cnbwb]';
B11=[0 -Izz\q*S*b*Cybt]';
B1a3=[B10 B11]*sigmabeta;

B20=[0 0]';
B21=[0 -Izz\q*S*b*Cybt]';
B2a3=[B20 B21];

% initial parameters
Vto=0.47;             % original tail volume
Vro=0.3*Vto;          % original relative rudder volume
zeta=[Vto]';
weight=[1 ];

% sigmabeta is variance on beta disturbances
B1=[0;-Izz\q*S*b*(Cnbwb+Cybt*Vto)]*sigmabeta;
C=[1/0.035 0; 0 0];      % weights rms beta to be less than 2 deg
D=[0;1/0.175];           % weights rms rudder deflection to be less than 10 deg
```

3. Example Three- Optimal Vertical Tail for Joint $\mathcal{H}_\infty$ Pole-Placement Specification

The Joint $\mathcal{H}_\infty$ Pole-Placement example problem used the identical set up script as example one above. The difference was that a different optimization function code was exercised (`plantop6` for the joint constraint in lieu of `plantopt2` for the pure $\mathcal{H}_\infty$ constraint).

4. Example Four- Optimal Vertical Tail for Joint $\mathcal{H}_\infty$ Static Moment Specification

Example Four used the identical set up script as example one above. The difference was that a different optimization function code was exercised (`plantop2a` for the joint constraint in lieu of `plantopt2` for the pure $\mathcal{H}_\infty$ constraint).

5. Example Five/Six- Plant and Controller Optimization with Maneuvering Constraints

The following script creates the bases for the longitudinal F-14 autoland problem. The aerodynamic derivative data was extracted from [Ref. 48], and the decomposition into the wing/body and horizontal tail contributions is in accordance with [Ref. 40]. The coding of the flight dynamics equations was verified by a cross check of the nominal plant ($\zeta = [1,\ 1]^T$) with a linear plant extracted from a linearization of the nonlinear equations used for the design examples of Chapter IV. This routine does provide the bases for uncertainty modeling, though this was not successfully demonstrated in an example problem.

This model does *not* include actuator dynamics. Predecessors did include actuator dynamics, but problems were encountered in solving the initial feasibility problem using either Riccati or interior point methods. These problems were attributed to the stiff geometry that the actuator poles introduced. See the script for further comments regarding its use.

The plant optimization function `plantopt2` was used for Example Five, while `plantopt2a` was used for Example Six. In the later two cases, the maneuverability constraint was passed into the optimization function through the input variable `Da`.

One aspect of this physical example is contrived. Specifically, the thrust was fixed for the problem and not used as a control input. This was necessary in the absence of actuator models, for otherwise the optimization routine shifted all the control energy into thrust in order to collapse the aerodynamic surface sizes.

An important practical issue relevant to the subject of the general problem's geometry was observed during these experiments. The method of centers code `centers3` has a numerical input argument `prec` which determines the termination criteria for the Newton search for the analytic center. It essentially determines how deep the search must go before it is considered to be "close enough" to the optimal value. In a strictly affine problem `prec` is related to the precision of the final output relative to the optimal value. The default for this value had been set to 10^{-3}, which would ensure that the output was within 0.1of the optimal value. Because of our methodology in employing the method of centers in alternating directions across the feasible set, this was no longer true. The output for the above problems could vary as much as a factor of 20 if `prec` was not set low enough. As `prec` was adjusted from 10^{-3} to 10^{-6}, the total cost at the conclusion of one set of trials improved by a factor of nearly 20. No further improvement was then noted as `prec` was adjusted from 10^{-6} to 10^{-9}.

f14 opt 3.m

```
% Establishes plant matrices for flight path control problem with
% stab and DLC control power adjustable.
% Thrust and actuators have been removed.

% This problem is hinf feasible, but not jointly hinf and pole-placement
% feasible for (87.3,1) circle.
% Converges successfully with plantopt2 (hinf only).
% The precision argument for plantopt2 must be set to 1e-6 or smaller
% to find the feasible controller.
```

```
g=32.174;                % fps
m=54000/g;               % slugs
Iyy=247194;              % slug-ft^2
S=565 ;                  % ft^2
cbar=9.8;                % ft
U=134*1.6889;            % fps
M=.203;
p=2116;                  % lbf/ft^2
Q=.7*p*M*M;              % lbf/ft^2
alpha=11.4/57.3;         % rad
u0=cos(alpha)*U;
th0=alpha;               % rad - trim point is level flight
sigma_cmd=3/57.3;        % rad
sigma_alpha=5/U;         % rad
omega1=40;               % rad/s  - Actuator bandwidths
omega2=80;               % rad/s
omega3=5;                % rad/s
rho1=0.1;                % DLC drag to lift ratio
rho2=0;                  % DLC moment to lift ratio

% F-14 stability derivative data
CLt=1.49534913;
CDt=0.37405269;
CMt=0;
CDU=0;
CLU=0;
CMU=0;
CDA=0.0208*57.3;         % rad^-1
CLA=0.0799*57.3;         % rad^-1
CMA=-0.0115*57.3;        % rad^-1
CLQ=5.45;                % rad^-1
CMQ=-14.3;               % rad^-1
CLAD=-0.51;              % rad^-1
CMAD=-0.93;              % rad^-1

CLIS=0.0141*57.3;        % rad^-1
CMIS=-0.0201*57.3;       % rad^-1

% Scale initial DLC for -0.1 g accel at full deflection of 0.5 rad
CLDLC=-0.1*CLt/0.5;      % rad^-1
CDDLC=rho1*CLDLC;        % rad^-1
CMDLC=rho2*CLDLC;

% Determine the wing body derivatives:
xc=-CMIS/CLIS;           % ratio of tail lever arm to mean chord
CDAwb=CDA;
CLAwb=CLA-CLIS;
CMAwb=CMA-xc*CLIS;
CLQwb=CLQ-2*xc*CLIS;
CMQwb=CMQ+2*xc^2*CLIS;
CLADwb=CLAD+2*xc*CLIS;
```

```
CMADwb=CMAD-2*xc^2*CLIS;

% determine Rotation Matrix from Lift/Drag to body axis
T=[-cos(alpha) sin(alpha) 0;-sin(alpha) -cos(alpha) 0;0 0 cbar];
QT=Q*S*T;
c2U=cbar/2/U;

% Determine Corrected Inertia Matrix
J=diag([m*U,m*U,Iyy,1])-[c2U*QT*[0 0 0;0 CLAD 0;0 CMAD 0 ],[0;0;0];0 0 0 0];

% Build aero blocks of state matrices
% states are u,alpha,q,theta,alpha/s,gamma_er/s
a00=[2*CDt -CLt 0;2*CLt CDt 0; 2*CMt 0 0];
a01=[CDU/U CDAwb 0; CLU/U CLAwb CLQwb*c2U;CMU/U CMAwb CMQwb*c2U];
a0=[QT*(a00+a01),-m*g*[cos(th0);sin(th0);0]; 0 0 1 0];
a0(1:2,3)=m*U*[-sin(alpha);cos(alpha)]+a0(1:2,3);
a1=[QT*CLIS*[0 0 0;0 1 2*xc*c2U; 0 xc -2*xc^2*c2U],[0;0;0];0 0 0 0];
a2=zeros(4,4);
a0=J\a0;                    a1=J\a1;

% control inputs are stab(rad),DLC (rad), thrust(lbf)
b0=zeros(4,2);
b1=J\[QT*[0 0;CLIS 0;CMIS 0];0 0];
b2=J\[QT*[0 CDDLC;0 CLDLC ;0 CMDLC ];0 0 ];

% Build system matrices
A0=zeros(6,6);   A1=A0;  A2=A0;

A0(1:4,1:4)=a0;                A1(1:4,1:4)=a1;                A2(1:4,1:4)=a2;
A0(5,2)=1;
A0(6,[2,4])=[-1 1];

Aa=[A0 A1 A2];

% exogenous inputs alpha_dist, gamma_cmd
B10= zeros(6,2);   B11=B10;   B12=B10;

B10(1:4,1)=a0(:,2)*sigma_alpha;           B10(6,2)=-sigma_cmd;
B11(1:4,1)=a1(:,2)*sigma_alpha;
B12(1:4,1)=a2(:,2)*sigma_alpha;

B1a=[B10 B11 B12];

B20= [b0; 0 0;0 0];     B21=[b1; zeros(2,2)];       B22=[b2; zeros(2,2)];

B2a=[B20 B21 B22];   % factor required to achieve hinf feasibility

% Build Output matrices
%
% z=[stab,DLC,alpha/s,gamma_err/s,alpha]
%
```

```
C=zeros(5,6);   D1=zeros(5,2);  D2=eye(5,2);
C(3:4,5:6)=eye(2);
C(5,2)=1;

% Output criteria:
% stab<20 deg
% DLC <40 deg
% alpha_err< 1.5 deg

zScale=diag([57.3/20,57.3/40,.00001,.00001,57.3/1.5]);
C=zScale*C;      D1=zScale*D1;    D2=zScale*D2;

zeta=[10;10];
wgt=[3,1];

% Build system matrices for uncertainty inputs and outputs
Delta=0.1*eye(6,3);                % Scaling for 10% uncertainty
B3=Delta*QT;
C3_0=QT\J(1:3,1:3)\QT*[a01,zeros(3,3)];
C3_1=[QT\a1(1:3,:),zeros(3,2)];
C3a=[C3_0,C3_1,zeros(3,6)];
D3a=QT\B2a(1:3,:);

% Add open loop requirement to maintain a max neg. pitch rate
% of 0.2 rad/sec for elev. deflection from trim of -20 deg

de_max = -20;
q_max = 0.2;

E0 = A0(3,3)*q_max + B20(3,1)*de_max/57.3;
E1 = A1(3,3)*q_max + B21(3,1)*de_max/57.3;
E2 = 0;
Ea = [E0 E1 E2];


% end f14_opt_3
```

6. Example Seven- Plant and Controller Optimization with Maneuvering Constraints and Directed Thrust

The following script is nearly identical to the script above, with the exception of the inclusion of directed thrust. The directed thrust shows up principally in the B_2 matrix where it influences both the closed-loop controller, and the open-loop maneuverability constraint. The maneuverability specification is identical to that

applied for Example Six.

f14 opt 4

```
% Establishes plant matrices for flight path control problem with
% stab and DLC control power adjustable.
% Thrust and actuators have been removed.

% This problem is hinf feasible, but not jointly hinf and pole-placement
% feasible for (87.3,1) circle.
% Converges successfully with plantopt2 (hinf only).
% The precision argument for plantopt2 must be set to 1e-6 or smaller
% to find the feasible controller.


g=32.174;                  % fps
m=54000/g;                 % slugs
Iyy=247194;                % slug-ft^2
S=565 ;                    % ft^2
cbar=9.8;                  % ft
U=134*1.6889;              % fps
M=.203;
p=2116;                    % lbf/ft^2
Q=.7*p*M*M;                % lbf/ft^2
alpha=11.4/57.3;           % rad
u0=cos(alpha)*U;
th0=alpha;                 % rad - trim point is level flight
sigma_cmd=3/57.3;          % rad
sigma_alpha=5/U;          % rad
omega1=40;                 % rad/s  - Actuator bandwidths
omega2=80;                 % rad/s
omega3=5;                  % rad/s
rho1=0.1;                  % DLC drag to lift ratio
rho2=0;                    % DLC moment to lift ratio

% F-14 stability derivative data
CLt=1.49534913;
CDt=0.37405269;
CMt=0;
CDU=0;
CLU=0;
CMU=0;
CDA=0.0208*57.3;           % rad^-1
CLA=0.0799*57.3;           % rad^-1
CMA=-0.0115*57.3;          % rad^-1
CLQ=5.45;                  % rad^-1
CMQ=-14.3;                 % rad^-1
CLAD=-0.51;                % rad^-1
CMAD=-0.93;                % rad^-1

CLIS=0.0141*57.3;          % rad^-1
```

```
CMIS=-0.0201*57.3;          % rad^-1

% Scale initial DLC for -0.1 g accel at full deflection of 0.5 rad
CLDLC=-0.1*CLt/0.5;         % rad^-1
CDDLC=rho1*CLDLC;           % rad^-1
CMDLC=rho2*CLDLC;

% Determine the wing body derivatives:
xc=-CMIS/CLIS;              % ratio of tail lever arm to mean chord
CDAwb=CDA;
CLAwb=CLA-CLIS;
CMAwb=CMA-xc*CLIS;
CLQwb=CLQ-2*xc*CLIS;
CMQwb=CMQ+2*xc^2*CLIS;
CLADwb=CLAD+2*xc*CLIS;
CMADwb=CMAD-2*xc^2*CLIS;

% determine Rotation Matrix from Lift/Drag to body axis
T=[-cos(alpha) sin(alpha) 0;-sin(alpha) -cos(alpha) 0;0 0 cbar];
QT=Q*S*T;
c2U=cbar/2/U;

% Determine Corrected Inertia Matrix
J=diag([m*U,m*U,Iyy,1])-[c2U*QT*[0 0 0;0 CLAD 0;0 CMAD 0 ],[0;0;0];0 0 0 0];

% Build aero blocks of state matrices
% states are u,alpha,q,theta,alpha/s,gamma_er/s
a00=[2*CDt -CLt 0;2*CLt CDt 0; 2*CMt 0 0];
a01=[CDU/U CDAwb 0; CLU/U CLAwb CLQwb*c2U;CMU/U CMAwb CMQwb*c2U];
a0=[QT*(a00+a01),-m*g*[cos(th0);sin(th0);0]; 0 0 1 0];
a0(1:2,3)=m*U*[-sin(alpha);cos(alpha)]+a0(1:2,3);
a1=[QT*CLIS*[0 0 0;0 1 2*xc*c2U; 0 xc -2*xc^2*c2U],[0;0;0];0 0 0 0];
a2=zeros(4,4);
a0=J\a0;                    a1=J\a1;

% control inputs are stab(rad),DLC (rad), thrust(lbf)
b0=zeros(4,2);
b1=J\[QT*[0 0;CLIS 0;CMIS 0];0 0];
b2=J\[QT*[0 CDDLC;0 CLDLC ;0 CMDLC ];0 0 ];

% create directed thrust control input
T0 = 13118;
thetae0 = 0/57.3;
Px = xc + 3;        % thrust lever arm

Bdt = zeros(6,1);
Bdt(1) = -T0*sin(thetae0)/m;
Bdt(2) = -T0*cos(thetae0)/(m*U);
Bdt(3) = -Px*T0*cos(thetae0)/m;

% Build system matrices
```

```
A0=zeros(6,6);   A1=A0;  A2=A0;

A0(1:4,1:4)=a0;                A1(1:4,1:4)=a1;              A2(1:4,1:4)=a2;
A0(5,2)=1;
A0(6,[2,4])=[-1 1];

Aa=[A0 A1 A2];

% exogenous inputs alpha_dist, gamma_cmd
B10= zeros(6,2);   B11=B10;   B12=B10;

B10(1:4,1)=a0(:,2)*sigma_alpha;          B10(6,2)=-sigma_cmd;
B11(1:4,1)=a1(:,2)*sigma_alpha;
B12(1:4,1)=a2(:,2)*sigma_alpha;

B1a=[B10 B11 B12];

B20= [b0; 0 0;0 0];
B20(:,1) = B20(:,1) + Bdt;
B21=[b1; zeros(2,2)];
B22=[b2; zeros(2,2)];

B2a=[B20 B21 B22];   % factor required to achieve hinf feasibility

% Build Output matrices
%
% z=[stab,DLC,alpha/s,gamma_err/s,alpha]
%
C=zeros(5,6);   D1=zeros(5,2);  D2=eye(5,2);
C(3:4,5:6)=eye(2);
C(5,2)=1;

% Output criteria:
% stab<20 deg
% DLC <40 deg
% alpha_err< 1.5 deg

zScale=diag([57.3/20,57.3/40,.00001,.00001,57.3/1.5]);
C=zScale*C;      D1=zScale*D1;    D2=zScale*D2;

zeta=[10;10];
wgt=[3,1];

% Build system matrices for uncertainty inputs and outputs
Delta=0.1*eye(6,3);                % Scaling for 10% uncertainty
B3=Delta*QT;
C3_0=QT\J(1:3,1:3)\QT*[a01,zeros(3,3)];
C3_1=[QT\a1(1:3,:),zeros(3,2)];
C3a=[C3_0,C3_1,zeros(3,6)];
D3a=QT\B2a(1:3,:);
```

```
% Add open loop requirement to maintain a max neg. pitch rate
% of 0.5 rad/sec for elev. deflection from trim of -20 deg
% and a thrust deflection of 5 degrees at 20 Klbs

de_max = -20;
q_max = .5;
thetae_max = -20;

E0 = A0(3,3)*q_max + B20(3,1)*de_max/57.3 + Bdt(3)*thetae_max/57.3;
E1 = A1(3,3)*q_max + B21(3,1)*de_max/57.3;
E2 = 0;
Ea = [E0 E1 E2];

theta = 0.1; gam  = 1; prec = 1e-6; thres = 1e-9;

[z2,xi,K,Z,Rc,Tc,J2,err]=plantopt2a(Aa,B1a,B2a,C,D2,Ea,zeta,wgt,thres,gam,theta,prec);
% end f14_opt_4
```

REFERENCES

1. MIL-F-8785C, *Military Specification: Flying Qualities of Piloted Aircraft*, November, 1980.

2. MIL-F-1797, *Military Specification: Flying Qualities of Piloted Aircraft*, 1990.

3. Foster, J. and others,"NASA/Navy Investigation of Lateral-Directional Control Requirements for High Performance Fighter/Attack Aircraft," AIAA Guidance, Navigation, and Control Conference, Monterey, CA, August 1993.

4. NASA Technical Memorandum 101684, *Development of a Preliminary High-Angle-of-Attack Nose-Down Pitch Control Requirement for High-Performance Aircraft* by L. Nguyen and J. Foster, Langley Research Center, 1990.

5. Air Force Wright Aeronautics Laboratories Report AFWAL-TR-87-3018, *Control Power Requirements for Reduced Static Stability Aircraft*, by H. Beaufrere and others, Grumman Corporation, June 1987.

6. Kaminer, I., P. P. Khargonekar, and G. Robel,"Design of localizer capture and track modes for a lateral autopilot using $\mathcal{H}_\infty$ synthesis," *IEEE Control Systems Magazine*, **10**: 13–21, 1990.

7. Kaminer, I. and P.P. Khargonekar, " Design of Flare Control Law for Longitudinal Autopilot Using $\mathcal{H}_\infty$ Synthesis," *Proc. of the 29th Conference on Decision and Control*, Honolulu, 2981-2986, 1990.

8. Kaminer, I., A. M. Pascoal, C. Silvestre, and P. P. Khargonekar, "Design of a control system for an underwater vehicle using $\mathcal{H}_\infty$ synthesis," *Proceedings of the 30th Conference on Decision and Control*, England, 2350–2355, December 1991.

9. Sivashankar, N., I. Kaminer, and P. P. Khargonekar, "Design of a Controller for the Turret System using $\mathcal{H}_\infty$ Synthesis," *Proceedings of the American Control Conference*, San Francisco, 1612–1616, 1993.

10. Sivashankar, N., I. Kaminer, and D. Kuechenmeister, "Design, Analysis and Hardware-in-the-Loop Simulation of a MIMO Controller for a VTOL Unmanned Aerial Vehicle using $\mathcal{H}_\infty$ Synthesis," preprint, September 1993.

11. Khargonekar, P. and M. Rotea, "Mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ control: A convex optimization approach," *IEEE Trans. on Automatic Control*, **36**, 824-837, 1991.

12. Kaminer, I., P. Khargonekar, and M. Rotea, Mixed $\mathcal{H}_2$ / $\mathcal{H}_\infty$ control for discrete time systems via convex optimization. *Automatica*, **29**, 50-70,1993.

13. Boyd, S. and C. Barratt, *Linear Controller Design*. Prentice-Hall, New Jersey, 1990.

14. Boyd, S. and L. El Ghaoui, "Method of Centers for Minimizing Generalized Eigenvalues", preprint, February 1993.

15. Boyd, S., L. El Ghaoui, E. Feron, and V.Balakrishan, *Linear Matrix Inequalities in Systems and Control Theory*, June 1993 draft.

16. Nesterov, Y. and A. Nemirovsky. *Interior Point Polynomial Methods in Convex Programming: Theory and Applications*, SIAM, 1993.

17. J. C. Doyle, B.A. Francis, and A.R. Tannenbaum *Feedback Control Theory*, 1992.

18. Doyle, J. C., K. Glover, P. P. Khargonekar, and B. A. Francis, "State-space solutions to standard $\mathcal{H}_2$ and $\mathcal{H}_\infty$ and control problems," *IEEE Transactions on Automatic Control,* **34**, 831-847, 1989.

19. Glover, K. and J. C. Doyle, "State-Space formulas for all stabilizing controllers that satisfy an $\mathcal{H}_\infty$ norm bound and relations to risk sensitivity", *Systems & Control Letters* **11**, 167-172, 1988.

20. Khargonekar, P. P., I. R. Petersen, and M. A. Rotea, " $\mathcal{H}_\infty$ optimal control with state feedback", *IEEE Transactions on Automatic Control*, AC-33(8):768-788, 1988.

21. Khargonekar, P. P., I. R. Petersen, and K. Zhou, "Robust stabilization of uncertain linear systems: Quadratis stabilizability and $\mathcal{H}_\infty$ control theory", *IEEE Transactions on Automatic Control*, AC-35(3): 356-361, 1990.

22. Petersen, I. R., "Disturbance attenuation and $\mathcal{H}_\infty$ optimization: A design method based on Riccati equation", *IEEE Transactions on Automatic Control*, AC-32(5):427-429, 1987.

23. Laub, A. , "A Schur method for solving algebraic Riccati equations", *IEEE Transactions on Automatic Control*, AC-24(6):913– 921, March 1979.

24. The Mathworks, Inc., *μ-Analysis and Synthesis Toolbox: User's Guide*, by G. Balas and others, 1991.

25. Kaminer, I. *Application of $\mathcal{H}_\infty$ Synthesis to the Motion Control of Rigid Bodies and Related Theory*, Ph.D. Dissertation, University of Michigan, 1992.

26. Bernussou, J. , P.L.D. Peres, and J.C. Jeromel, "A Linear Programming Oriented Procedure for Quadratic Stabilization of Uncertain Systems," *Systems Control Letters*, **13**:65-72, 1989.

27. Gahinet, P., "A convex Characterization of Parameter-Dependent $\mathcal{H}_\infty$ Controllers," *31st IEEE Conference on Decision and Control*, Tucson, 937-942, 1992.

28. Rotea ,M. and P. Khargonekar, $\mathcal{H}_2$ optimal control with an $\mathcal{H}_\infty$ constraint: the state-feedback case. *Automatica*, **27**, 307-316, 1991.

29. Rotea, M., The generalized $\mathcal{H}_2$ control problem. *Automatica*, **29**, March 1993.

30. Rotea, M. and P. Khargonekar, In: *Proc. of the 30th Conference on Decision and Control*, Brighton, UK, 2719-2720, 1991.

31. Stoorvogel, A., *The $\mathcal{H}_\infty$ control problem: A State-Space Approach,* Ph.D. Dissertation, University of Eindhoven, 1990.

32. Furuta, K. and S. B. Kim,"Pole assignment in a specified disk," *IEEE Transactions on Automatic Control*, AC-32:423-427, 1987.

33. Sivashankar, N., I. Kaminer, and P. P. Khargonekar, "Optimal Controller Synthesis with $\mathcal{D}$ Stability", to appear in *Automatica*, 1994.

34. Doyle, J.C., "Analysis of Feedback Systems with Structured Uncertainties," *IEEE Proc.*, **129**, 1982.

35. Doyle, J.C., "Structured Uncertainty in Control System Design", *Proceedings of the 24th Conference on Decision and Control*, Fort Lauderdale, FL, 260-265.

36. Desoer, C.A. and M. Vidyasagar, *Feedback Systems: Input-Output Properties*, Academic Press, 1975.

37. Roskam, J., *Airplane Flight Dynamics and Automatic Flight Controls*, Roskam Aviation and Engineering Corporation, Lawrence, Kansas, 1979.

38. Kailath, T. *Linear Systems*, Prentice Hall, 1980.

39. Kaminer, I., A. M. Pascoal, and P. P. Khargonekar, "A Velocity Algorithm for the Implementation of Gain Scheduled Controllers", preprint, August 1993.

40. Hoak, D. E. and others, *USAF Stability and Control DATCOM*, April 1978.

41. Cottle, R.W. "Manifestations of the Schur Complement" *Linear Algebra and its Applications*, **8**, 189-211, 1974.

42. Packard, A. and others, "A Collection of Robust Control Problems Leading to LMI's," *Proceedings of the 30th Conference on Decision and Control*, December 1991, pp. 1245-1250.

43. Becker, G. and A. Packard, robust Performance of Linear, Parametrically Varying Systems using Parametrically-Dependent Linear, dynamic Feedback," *Systems and Control Letters*, February 1994.

44. Knotts, L. H. and others, "Aircraft Control System Rate Limiting", *Cockpit*, Society of Experimental Test Pilots, 1993.

45. Polak, E. and Y. Wardi,"Nondifferentiable Optimization Algorithm for Designing Control Systems Having Singular Value Inequalities," *Automatica*,**18(3)**, 267-283, 1982.

46. Geering, H., "On Calculating Gradient Matrices," *IEEE Transactions of Automatic control*, 615, August 1976.

47. Iglesias, P.A. and K. Glover, "State Space Approach to Discrete Time $\mathcal{H}_\infty$ Control," Cambridge University, May 1990.

48. Grumman Aerospace Corporation Report A55-35-R-91-06, *F-14D Pre-Deployment Upgrade Stability and Control and Flying Qualities Report*, by H.W. Stiles and others, pp. 4-8 and 4-12, December 1991.

INITIAL DISTRIBUTION LIST

		No. of Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Dr. Isaac I. Kaminer Department of Aeronautics and Astronautics, Code AA/KA Naval Postgraduate School Monterey, CA 93943-5000	5
4.	Dr. Robert J. Niewoehner 44176 Mimosa Lane California, MD 20619	2
5.	Chairman Department of Aeronautics and Astronautics Naval Postgraduate School Monterey, CA 93943-5000	2
6.	Dr. William Gragg Department of Mathematics, MA/Gr Naval Postgraduate School Monterey, CA 93943-5000	1
7.	Dr. Roberto Cristi Department of Electrical and Computer Engineering, Code EC/Cx Naval Postgraduate School Monterey, CA 93943-5000	1

No. of Copies

8. Dr. Pramod Khargonekar 1
Department of Electrical Engineering
and Computer Science
The University of Michigan
Ann Arbor, MI 48109-2122